Received: 11 May 2025 | Revised: 18 Aug 2025 | Accepted: 1 Jun 2026 | Published: 30 Jun 2026
Volume 4(1) Pages 128-149 DOI: https://doi.org/10.56279/jicts.v4i1.408

JICTS
Journal of ICT Systems

Comparative Vulnerability Assessment of Android and iOS Mobile Banking Applications in Tanzania

Daniel Ngondya[1]

Department of Computer Science and Engineering, The University of Dodoma, Dodoma, Tanzania

[1]Corresponding author
Email: daniel.ngondya@udom.ac.tz

Funding information

No external funding was received for this work.

Abstract

As mobile banking usage surges, the risk of exploitation by malicious actors also increases, taking advantage of vulnerabilities in applications. Vulnerabilities on the mobile banking applications may differ due to platform-specific architectures, development practices, and the degree of enforcement of data protection regulations. We compare vulnerabilities of the applications of Tanzania's commercial banks based on Android and iPhone Operating System (iOS) platforms. A static approach to vulnerability assessment was employed by scanning source code files of the applications using a Mobile Application Security Verification Standard (MASVS) compliant tool, the Mobile Security Framework (MobSF). The results show that, on average, iOS applications have a larger number of high-risk vulnerabilities at 2.3 than Android applications at 2.1. Android averages at 11.4 per application for medium risk vulnerabilities compared with iOS' 1.5. It is further observed that Android has a higher variability of both high-risk and medium-risk vulnerabilities than iOS applications, and can work on vulnerable, unpatched, and unsupported Android versions below Android 9.

Keywords

Android iOS
Mobile banking applications
Mobile security framework
Static analysis
Vulnerability assessment

1. Introduction

Mobile device usage, especially Smartphones, is rising globally, with projections indicating 85% of people will own one by 2025 [1]. In Tanzania, 25.4 million Smartphone users were expected by 2024 [2]. Gender and income influence adoption, and smartphones are the primary means of Internet access in Tanzania, with 48 million subscriptions reported [3, 4]. Smartphones, via third-party mobile applications and internet connectivity, are utilised across various sectors, such as transportation, tourism, and finance [5, 6]. These applications gather extensive personal and sensitive data, including user accounts, location, and demographic details, to deliver personalised services and marketing [7-9].

Android and iOS are the most dominant smartphone operating systems, boasting up to 99.22% of the Smartphone market [10]. The dominance of these operating systems makes them a primary target for malicious actors. Additionally, vulnerabilities associated with these platforms have a significant impact on businesses and individuals. Android's open-source software development promotes customisation and lowers development costs, while iOS's proprietary nature optimises performance with specific hardware [11]. These differences result in varying vulnerabilities between the two platforms. Security and privacy risks arise from operating system vulnerabilities, particularly those linked to permission models, where apps may request excessive permissions or fail to obtain user consent [12]. Both Android and iOS devices with vulnerable kernels face attack risks, and weaknesses in memory management and user authentication can further jeopardise mobile application security.

Attackers target mobile applications, especially in finance, due to potential financial gain [13, 14]. Tanzanians increasingly use mobile banking, with around 30 million subscriptions [15]. Research shows that income and education levels impact mobile banking adoption, enabling transactions like balance checks, fund transfers, bill payments, and airtime purchases [16, 17]. Mobile banking application attacks expose sensitive financial information, facilitating fraud. In 2024, Tanzania lost approximately 2.18 million US Dollars due to such fraud, with nearly half of the banks' losses tied to digital banking [18, 19].

Works on comparative assessment of mobile banking applications are characterised by a limited sample size of studied applications, generic scanning tools, privilege escalation of studied devices, and have considered different regulatory environments of banks. With encrypted traffic (e.g., HTTPS, TLS, and VPN), generic scanning tools such as Wireshark are not effective without decryption keys or Man-in-the-Middle attack approaches. Privilege escalation by rooting or jailbreaking Smartphones increases the risk of vulnerabilities and exploits as it waives existing security mechanisms [20]. Moreover, in some jurisdictions, rooting or jailbreaking is illegal as it facilitates intellectual property infringement [21]. Furthermore, jailbreaking or rooting is not possible on every version, especially the latest ones.

Security issues are context-specific, influenced by skills, culture, awareness, and regulations [22, 23]. Analyzing mobile banking applications' source codes for Android and iOS provides relevant insights into cybersecurity, secure coding practices, and data protection. Vulnerabilities on mobile banking applications tend to vary due to platform-specific architectures, development practices, and the extent to which data protection regulations are enforced. In Tanzania, the increasing adoption of mobile banking is raising concerns in terms of security and integrity of the mobile banking applications, because vulnerabilities can lead to unauthorised access, data breaches, and financial fraud. This work sought to conduct a comparative vulnerability assessment of 28 mobile banking applications based on Android and iOS operating systems for commercial banks operating in Tanzania. Specifically, this work sought to address the following research questions:

i. What is the prevalence of vulnerabilities in Android and iOS mobile banking applications?
ii. What are the categories of vulnerabilities associated with Android and iOS mobile banking applications?

Key contributions of our study include enhancing privacy and personal data protection compliance as per local and international regulations. Establishing the prevalence and categories of mobile banking application vulnerabilities raises awareness among developers, organizations, vendors, policymakers, and security professionals, promoting a secure mobile application ecosystem. The work is structured into sections: Section 2 reviews related literature and identifies research gaps, Section 3 outlines methodologies, Section 4 discusses results on mobile banking vulnerability types, and Section 5 concludes the study and suggests future research directions.

2. Related Works

Mobile applications are prone to different kinds of vulnerabilities. Several databases document common vulnerabilities for different kinds of software, including mobile applications. The Open Web Application Security Project (OWASP) is a prominent organisation documenting vulnerabilities associated with mobile applications by regularly listing the top 10 common vulnerabilities for mobile applications [24]. The current OWASP Mobile Top 10 ranks common vulnerabilities on mobile applications as shown in Table 1.

The previous OWASP Mobile Top 10 (OWASP-2016) ranked the following vulnerabilities as the most common: improper platform usage, insecure data storage, insecure communication, insecure authentication, insufficient cryptography, insecure authorisation, client code quality, code tampering, reverse engineering, and extraneous functionality [25]. Compared with the previous list, the current list features new kinds of vulnerabilities, such as inadequate privacy controls, and has some of the previous vulnerabilities merged; for example, insecure communication and insecure authorisation have been merged to form insecure communication. The listed vulnerabilities are not the only ones that mobile applications face. Moreover, the listed vulnerabilities are contextual; therefore, their prevalence varies from one place to another depending on the level of skills of developers, compliance enforcement, and awareness of the society in general. OWASP Mobile Top 10 has been used as a reference for vulnerability assessment of mobile applications in works such as that of Acharya et al. [26] and Alanda et al. [25].

Instead of relying on lists for vulnerability assessment, security standards like PCI-DSS and the Mobile Application Security Verification Standard (MASVS) offer more extensive coverage. PCI-DSS is mandatory for financial institutions handling credit card information, while MASVS focuses on evaluating mobile application security [28]. MASVS provides a broader guidance than PCI-DSS and OWASP Mobile Top 10, which overlooks platform-specific vulnerabilities [29]. Intended for developers, architects, and pentesters, MASVS categorizes security requirements, further explicated in Table A1, and is supported by the OWASP Mobile Application Security Testing Guide (MASTG), which aligns test cases with MASVS controls.

MASTG aligns its test cases with MASVS control groups by providing procedures identifying vulnerabilities that violate the requirements [30]. MASTG test cases are presented in terms of categories labelled as MASVS-STORAGE, MASVS-CRYPTO, MASVS-AUTH, MASVS-NETWORK, MASVS-PLATFORM, MASVS-CODE, and MASVS-RESILIENCE [31]. These categories align with MASVS control group requirements. Appendix B shows some of the test cases for the MASVS-CRYPTO category, under the MASTG guideline. Some of the works that have used the MASVS

framework for vulnerability assessments include those by Chiboora et al. [32], Mahmud et al. [33], and Anwar et al. [34]. MASVS has been used in this work as the basis for comparing vulnerabilities for Android and iOS-based mobile banking applications.

Several studies have focused on vulnerability assessments of mobile banking applications. Falade and Ogundele evaluated six UK digital banks, finding common issues like implicit intent for service and misconfigured intent filters using AndroBugs [35]. Bojjagani and Sastry identified 356 exploitable vulnerabilities in unspecified Android banking apps through static and dynamic analysis, particularly in cryptography [36]. Titiakarawongse et al. compared Thai and non-Thai banks, using tools like MobSF, noting vulnerabilities linked to reverse engineering and insufficient cryptography, with Thai banks showing more weaknesses than their non-Thai counterparts [37].

Bojjagani and Sastry [14] assesses vulnerabilities in both Android and iOS mobile banking applications, unlike many prior works focusing solely on Android. They employed static and dynamic analysis on 5 Android and 3 iOS applications from Indian banks, utilizing over 10

Table 1. Latest release of OWASP mobile top 10 [27].

Rank	Vulnerability	Description
1	Improper Credentials Usage	Hardcoding of credentials in a mobile application's source code or improper handling thereof.
2	Inadequate Supply Chain Security	This involves inserting malicious code into the codebase or introducing backdoors and spyware during the build process. Also, includes exploiting vulnerabilities in third-party libraries, Software Development Kits, or vendors.
3	Insecure Authentication/Authorisation	Faking or bypassing authentication mechanisms and privilege escalation when the mobile application is offline.
4	Insufficient Input/output Validation	Includes sanitisation and validation of user input or network data for the mobile application.
5	Insecure Communication	Security of data exchanged between a mobile application and a remote server, as it passes through the carrier network and the internet.
6	Inadequate Privacy Controls	Has to do with the protection of Personally Identifiable Information such as names, addresses, gender, health status, etc.
7	Insufficient Binary Protections	Mobile application binaries can be targeted for several reasons, such as stealing valuable secrets, commercial Application Programming Interface (API) keys, or hardcoded cryptographic keys.
8	Security Misconfiguration	Misconfiguration in security settings in terms of permissions and controls leads to unauthorised or unintended access.
9	Insecure Data Storage	Considers vulnerabilities related to improper handling of user credentials, weak encryption algorithms, poor storage mechanisms, and insufficient data protection.
10	Insufficient Cryptography	Vulnerabilities related to key management, weaknesses in encryption algorithms, side-channel attacks, and brute-force attacks.

scanning tools such as Apktool Analyzer and Burpsuite. Their findings highlighted insufficient transport-layer protection as a common vulnerability, with applications still being vulnerable to Man-in-the-middle attacks despite using HTTPS.

Existing research on mobile banking application vulnerabilities primarily focuses on Android despite significant iOS user numbers. Various vulnerability scanning tools produce differing reports, complicating comparisons between Android and iOS platforms' strengths and weaknesses. Additionally, studies often utilize small sample sizes, making broad generalizations about the banking industry or mobile app developer communities. Furthermore, examining applications across varied data protection compliance jurisdictions hinders the ability to extract actionable insights for developers, regulators, and banks. Furthermore, some of the existing works have used both static and dynamic analysis to study vulnerabilities. Static analysis uncovers vulnerabilities by extracting information about functions and libraries used in the source code [38]. Unlike static analysis, dynamic analysis studies application behaviour when it is being executed [39]. Dynamic analysis has limited coverage as it cannot cover all possible execution paths or scenarios, implying it is more ideal for the detection of malware and tracking behaviour [40].

Tracking the behaviour of applications and their related malicious activities is beyond the scope of this work. Lastly, tools used for scanning vulnerabilities have assumed the OWASP Mobile Top 10 and vulnerability databases as the reference for identifying vulnerabilities in mobile banking applications. The list changes from time to time depending on the trends of vulnerabilities; as a result, it does not comprehensively cover all potential vulnerabilities, particularly platform-specific vulnerabilities. This work sought to compare vulnerability profiles of all Android and iOS mobile banking applications in Tanzania, using a single MASVS-compliant scanning tool to ensure comprehensive coverage of vulnerabilities.

3. Methods

3.1 Vulnerability Assessment Approach

The iOS App Store Package (IPA) and Android Package (APK) files were extracted from downloaded mobile banking applications from the App Store and Play Store, respectively. Static analysis was used to analyse the extracted IPA and APK files. The applications were downloaded between September and November 2024.

3.2 Population, Sample Size and Sampling of Mobile Banking Applications

The Bank of Tanzania (BoT) oversees the licensing and regulation of financial institutions in Tanzania. As per BoT records, currently, 34 commercial banks operate in the country, with 29 providing mobile banking services [41]. Among these, 28 banks support both Android and iOS, while the African Banking Corporation Tanzania Limited exclusively offers services on the Android platform, as confirmed through website visits and searches in app stores[1][2][3].

As per Lakens [42] and Singh and Masuku [43], a census approach can be employed for small populations. Based on the approach, all Android and iOS mobile banking applications of 28 commercial banks were considered for static analysis. Table A3 and Table A4 in the Appendix indicate banks and associated names of Android and iOS mobile banking applications considered in this study.

3.3 Tools and Lab Setup

APK and IPA files were extracted using APK extractor for Android and iMazing for iOS

[1] https://tanzania.accessbankplc.co.tz/about-us/corporate-profile

[2] https://play.google.com/store/apps

[3] https://www.apple.com/app-store/

mobile banking applications. For Android, the applications were downloaded from the Google Play Store and installed on a Tecno POP 7 with Android 12.0. APK extractor was used to retrieve the APK files. For iOS, the applications were installed on an iPhone 12 Pro with iOS 18.4.1, and iMazing was installed on a Windows 11 Pro machine to extract the IPA files after connecting the iPhone via USB.

For static analysis of APK and IPA files, MobSF version 4.0.7 was used. MobSF is an open-source tool that supports both static and dynamic analysis of mobile applications. Furthermore, MobSF can perform analysis of both Android and iOS-based applications, unlike tools such as AndroBugs and Quick Android Review (QARK). Importantly, MobSF is compliant with the MASVS standard. Since tools differ in terms of report format and vulnerability detection efficiency, using a single tool for Android and iOS applications provides an objective way to compare their vulnerabilities. MobSF and Docker virtualisation environments were installed on a Windows 11 Pro Machine. The former enabled access to the MobSF tool via a Web Browser. APK and IPA files were scanned by uploading them to MobSF via a Graphical User Interface (GUI) accessed through the Web Browser, and a vulnerability report was generated for each file (APK and IPA files).

3.4 Data Collection and Analysis

Reports generated by MobSF categorise risk levels as "High", "Medium", "Low", and "Info" or "Secure", depending on the potential impact of the vulnerability. High-risk vulnerability is used to point out critical vulnerabilities that, when exploited, can cause severe security breaches [44]. Furthermore, high risk vulnerabilities are challenging to fix, and, therefore, takes more time than medium risk vulnerabilities [45]. Medium risk or "warning" refers to a potentially risky vulnerability that can cause notable damage, although not as critical as that of high-risk vulnerabilities.

Good or recommended practices are those labelled as "secure", while non-critical practices and potential misconfigurations that need to be reviewed are labelled as "info". Owing to the severity of vulnerabilities, for each generated report, the name of the mobile banking application, its version, together with the number, type of high and medium risk vulnerabilities, and the MASVS control group, were collected. Microsoft Excel was used to organise and analyse collected data. A descriptive analysis indicating types of vulnerabilities and their associated frequencies was conducted.

Moreover, a correlation analysis was conducted to compare vulnerabilities on Android and iOS mobile banking applications. Owing to the severity of the analysed vulnerabilities, the banks indicated in Table A3 and Table A4 were shuffled and coded as B1, B2, B3, …, and B28 to avoid targeted exploitation by malicious actors. Such an approach has been used in works by Chiboora et al. [32], Bassolé et al. [46], and Titiakarawongse et al. [37].

4. Results and Discussion

4.1 Prevalence of High and Medium Risk Vulnerabilities among Mobile Banking Applications

Figure 1 shows the number of high-risk vulnerabilities observed from mobile banking applications based on both Android and iOS platforms. From Figure 1, it can be observed that up to 5 (17.8%) iOS applications have no high-risk vulnerabilities compared with only 2 (7.1%) for Android applications. Some banks, such as B1, B9, B12, B15, B16, and B22, have an equal number of high-risk vulnerabilities for both iOS and for both Android and iOS-based mobile applications. The mobile banking application by Bank 25 has the highest number of vulnerabilities Android

applications. In the rest of the applications, Android applications have a higher number of vulnerabilities than iOS' applications, and vice versa. Android applications have an average of 2.1 high-risk vulnerabilities, while iOS ones have 2.3. There is a larger variation in high-risk vulnerabilities on Android applications (3.25) than on those of iOS (2.06). Correlation analysis with Pearson Correlation Coefficient (PCC) was conducted to determine if a high number of high-risk vulnerabilities on Android applications leads to a high number of high-risk vulnerabilities for iOS applications. Results indicate a PCC value of 0.592, suggesting a moderately positive correlation. That is, high values of high-risk vulnerabilities on Android applications mean high values of the same on iOS applications.

Figure 2 illustrates the number of medium-risk vulnerabilities for Android and iOS mobile applications. From the Figure, it can be observed that iOS applications have a smaller number of medium-risk vulnerabilities (average=1.5) compared with those based on Android (average=11.4). Furthermore, Figure 2 shows that variability in the number of medium-risk vulnerabilities is higher with Android (variance=52.39) than with iOS (variance=0.99) applications. From the correlation analysis, the PCC value of 0.4334 indicates that there is a weak relationship between the number of medium-risk vulnerabilities on Android applications and iOS ones.

Variability in the number of high and medium risk vulnerabilities within the two platforms across the banking industry suggests a wide gap in secure mobile application development in terms of knowledge, skills, and experience [47]. The earlier argument can be observed in both Android and iOS platforms, where, for example, the iOS application of bank B2 has no high-risk vulnerabilities. In contrast, that of bank B25 has six high-risk vulnerabilities (Figure 1). Furthermore, the correlation analysis indicates that both high- and medium-risk vulnerabilities on Android lead to high- and medium-risk vulnerabilities on iOS applications. The preceding observation makes sense as, in organisations, teams that develop Android applications are the ones that do the same for iOS applications. It means the knowledge, skills, and experience gap is reflected in both platforms.

Additionally, there is a growing practice of developing mobile applications for one platform and converting them into another platform to ensure timely deployment, cost savings, and consistent features [48]. However, the practice is likely to carry vulnerabilities from one codebase to another, as observed by the correlation of high- and medium-risk vulnerabilities between Android and iOS-based vulnerabilities. Instead of merely focusing on addressing functional requirements and User Interface and User Experience (UI/UX), tools for code conversion or generation have to adequately take into account the security of the resulting code.

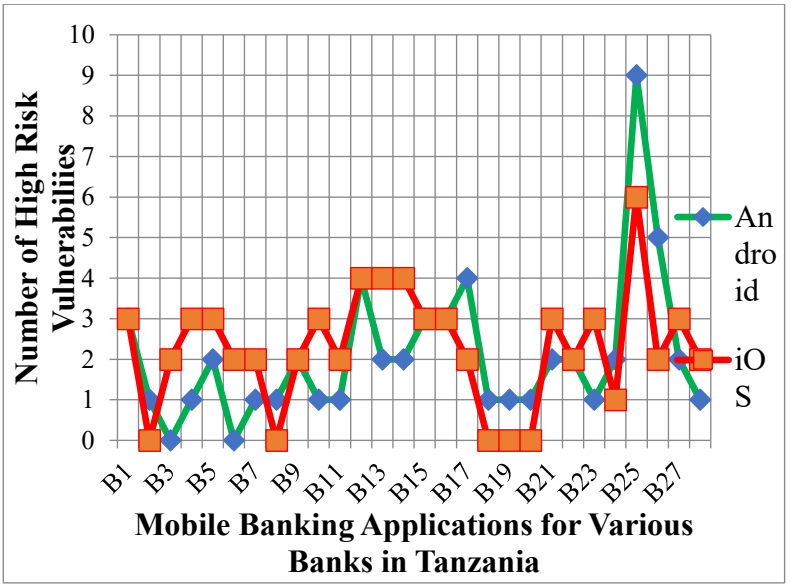


Figure 1. Comparison of the number of high-risk vulnerabilities on each of the Android and iOS applications.

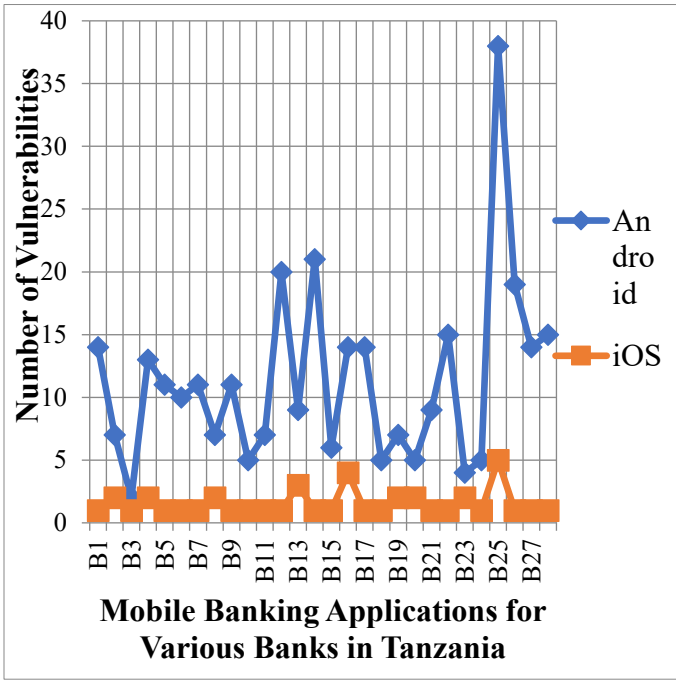


Figure 2. Comparison of the number of medium risk vulnerabilities on each of the android and iOS applications

4.2 Categories of High- and Medium-Risk Vulnerabilities among Mobile Banking Applications

The high-risk vulnerabilities for iOS applications have been ranked as indicated in Figure 3. From Figure 3, the use of insecure APIs, binary making use of the malloc function, and Application Transport Security allowing arbitrary loads are the top three vulnerabilities. Medium-risk vulnerabilities for iOS applications are indicated in Figure 4. From Figure 4, the three most common vulnerabilities are symbols stripped, NSRequiresCertificateTransparency Set to No, and Stack Canary.

Except for platform-specific vulnerabilities, MobSF automatically classified detected vulnerabilities according to MASVS control groups indicated in Table A1. In this case, the resulting categories were collected from MobSF reports generated. The MASTG was used to classify platform-specific vulnerabilities by matching vulnerability descriptions and testing guides such as those indicated in Table A2. TableA2 indicates test cases that apply for the cryptography category. Each of the MASVS control groups have their own test cases which were matched with description of platform-specific vulnerabilities. The resulting categories were integrated with those generated automatically by MobSF to provide comprehensive classification of vulnerabilities. Out of 8 categories of security requirements under MASVS (Table A1), iOS applications have only exhibited 3, namely code quality and build setting, network communication, and cryptography. This is illustrated by Figure 5 and Figure 6 for high-risk and medium-risk vulnerabilities, respectively. Figure 5 illustrates high-risk vulnerabilities that are categorised as being related to code quality and build setting, network communication, and cryptography, while Figure 6 shows medium-risk vulnerabilities associated with iOS applications, categorised as code quality and build setting, and network communication. In both Figure 5 and Figure 6, vulnerabilities under code quality and build setting are the most common, followed by the network communication category, and lastly is the cryptography category. Vulnerabilities under code quality and build setting for high-risk vulnerabilities include the following: making use of insecure Application Programming Interfaces (APIs), use of unsafe functions such as malloc, not adding a stack canary value to the stack, and not compiling the binary with the Automatic Reference Counting (ARC) flag.

The iOS applications' medium risk vulnerabilities related to code quality and build settings are: having Runpath Search Path (RPATH) set, having debug symbols stripped and thereby being unable to verify that stack canary is enabled or not, and the availability of debug symbols. Vulnerabilities related to network communication are insecure communication, such

as allowing insecure HTTP loads, disabling App Transport Security restrictions, disabling Certificate Transparency Protocol, and accepting vulnerable minimum Transport Layer Security Protocol (TLSv1.2). The vulnerability associated with the insufficient cryptography requirements is the use of insecure random number generators.

High-risk vulnerabilities for Android applications are indicated in Figure 7. Based on Figure 7, the three commonest high-risk vulnerabilities are ability to install the application on vulnerable and unpatched Android version, using the encryption mode CBC with PKCS5/PKCS7 padding which is vulnerable to padding oracle attacks, and any application on the mobile device being able to read a file because the file or SharedPreference is set to World-Readable. Figure 8 displays medium-risk vulnerabilities for Android applications, where the top three vulnerabilities are unprotected or inadequately protected broadcast receivers, services, and activities. In Figure 8, the Asterisks represent different types of broadcast receivers, services, and activities.

As for Android applications, high- and-medium risk vulnerabilities can be mapped into 6 out of 8 MASVS security requirements categories, as shown by Figure 9 and Figure 10, respectively. The categories include architecture, design and threat modelling, cryptography, data storage and privacy, Authentication and Session Management, code quality and build settings, and resilience. While there is an overlap of some categories for medium- and high-risk vulnerabilities, there is a difference in their prevalence. The most common high-risk category of MASVS security requirements is cryptography. Other categories include architecture, design and threat modelling, data storage and privacy, authentication and session management, and code quality and build setting (Figure 9). Figure 10 illustrates that the most common risk category for medium risk vulnerabilities is platform interactions, followed by data storage and privacy, code quality and build setting, cryptography, resilience, and architecture, design and threat modelling.

According to Figure 7, vulnerabilities associated with cryptography security requirements include use of CBC encryption mode with PKCS5/PKCS7 padding, which is prone to oracle padding attacks, use of a certificate algorithm that is vulnerable to collision attacks, use of encryption in ECB mode or calling Cipher.getInstance("AES"), which returns AES ECB mode by default, which is known to be a weak encryption algorithm, and the use of weak encryption or hashing algorithms such as SHA-1 or MD5. High-risk vulnerability associated with architecture, design and threat modelling category is that the mobile banking application can be installed on vulnerable and unpatched Android versions. High risk vulnerabilities related to the data storage and privacy category are: allowing public read or write access to file or SharedPreference-meaning any application can read or write to file or SharedPreference. An application's link asset uniform resource locator (URL) not being set or configured properly is a high-risk vulnerability related to the authentication and session management category.Enabling debugging of production builds or remote WebView is categorised as a vulnerability related to code and build settings. The network communication category of security requirements is related to insecure implementation of Secure Socket Layer (SSL), and insecure implementation of WebView to accept any certificate and ignore any SSL certificate errors.

With regard to medium-risk vulnerabilities for Android applications, those categorised as being related to platform interactions are ones associated with a lack of protection or insufficient

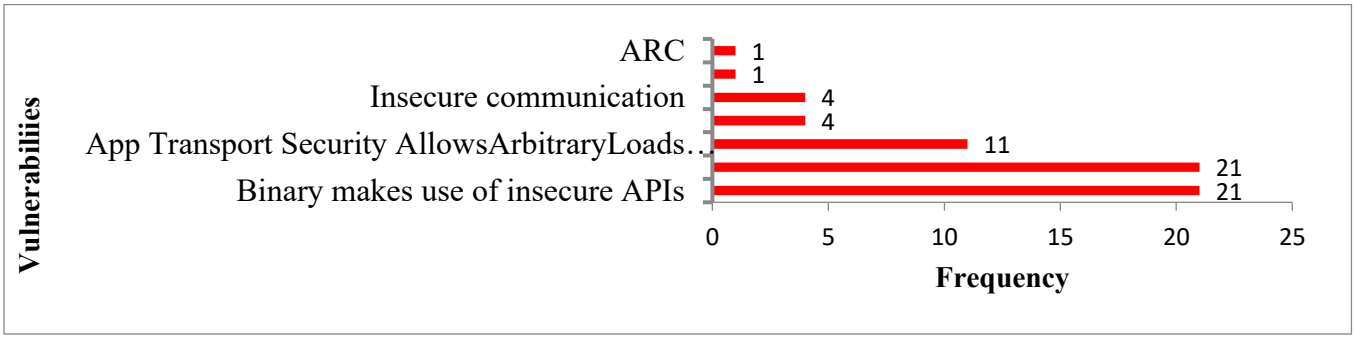


Figure 3. Illustration of the commonest high-risk vulnerabilities for iOS Applications.

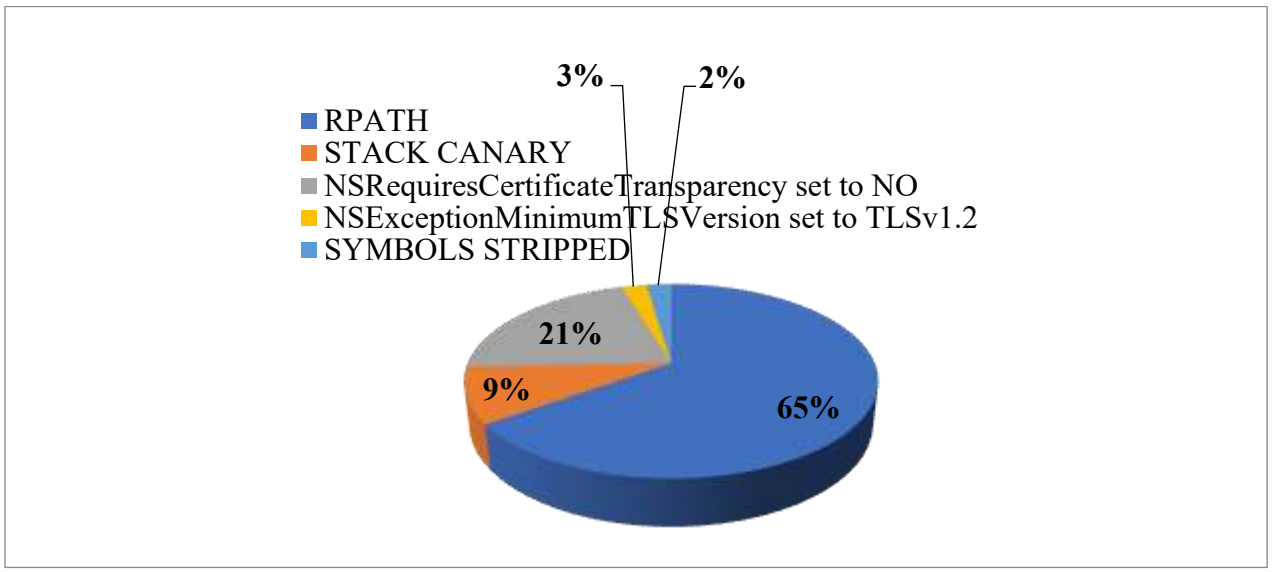


Figure 4. Medium Risk Vulnerabilities for iOS Applications.

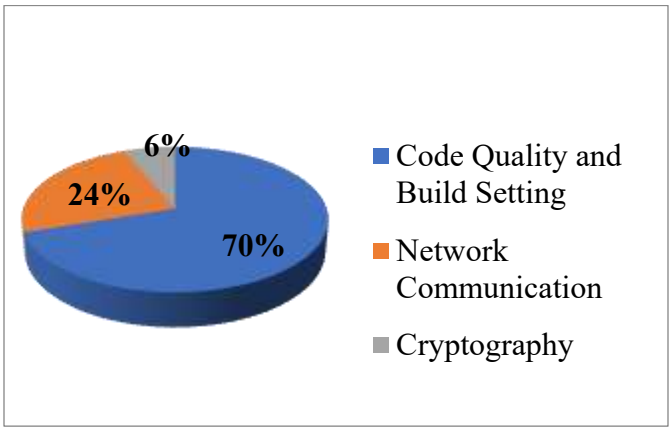


Figure 5. Categories of high-risk vulnerabilities as per the MASVS framework for iOS applications

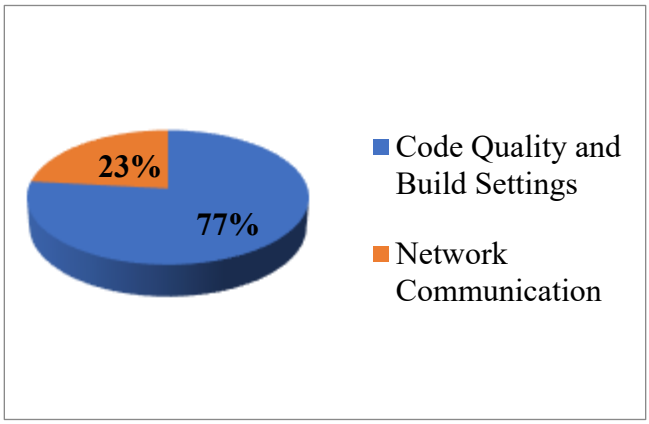


Figure 6. Categories of medium risk vulnerabilities as per the MASVS framework for iOS applications.

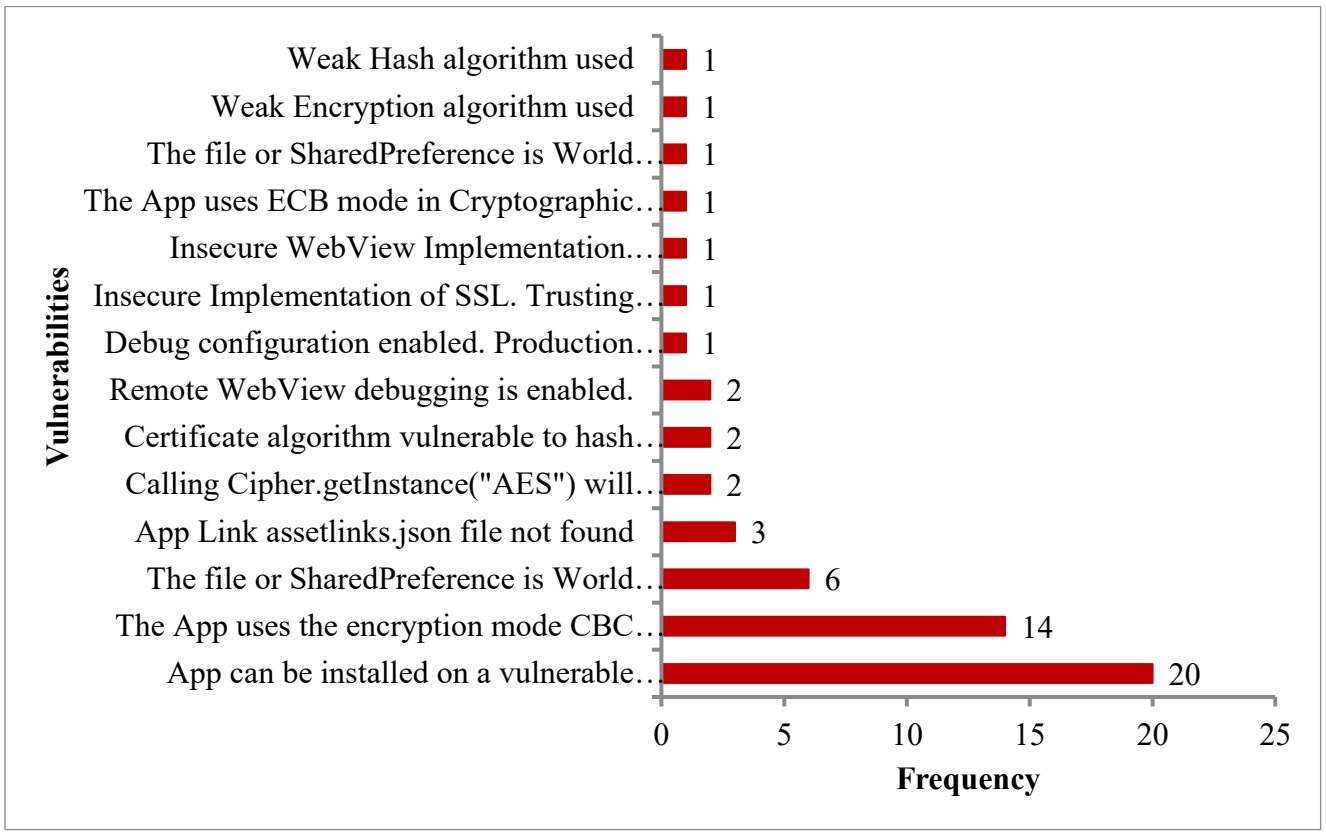


Figure 7. Illustration of the commonest high-risk vulnerabilities for android applications

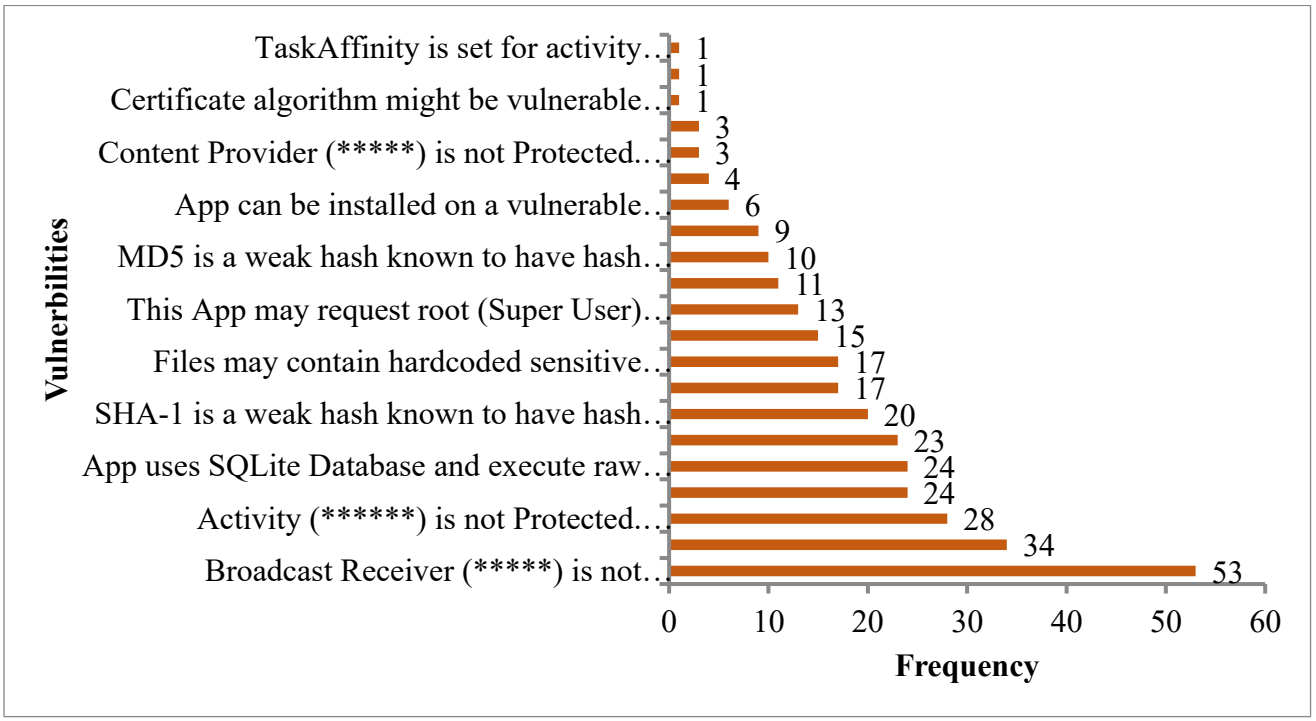


Figure 8. Medium risk vulnerabilities for iOS applications.

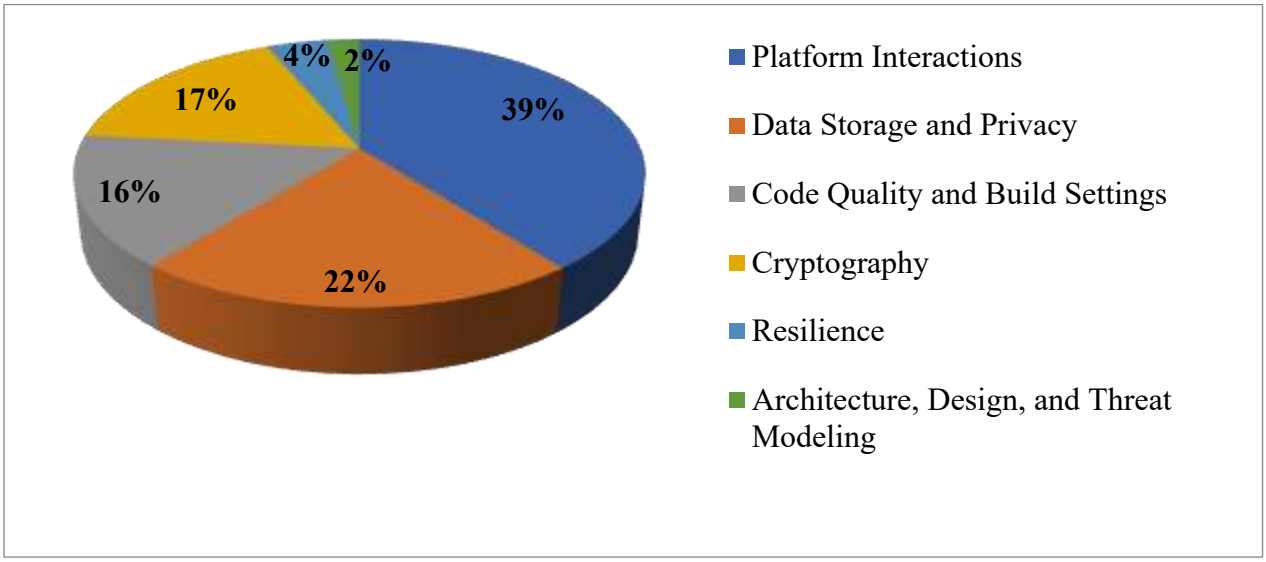


Figure 9. Categories of Medium Risk Vulnerabilities as per the MASVS framework for Android Applications.

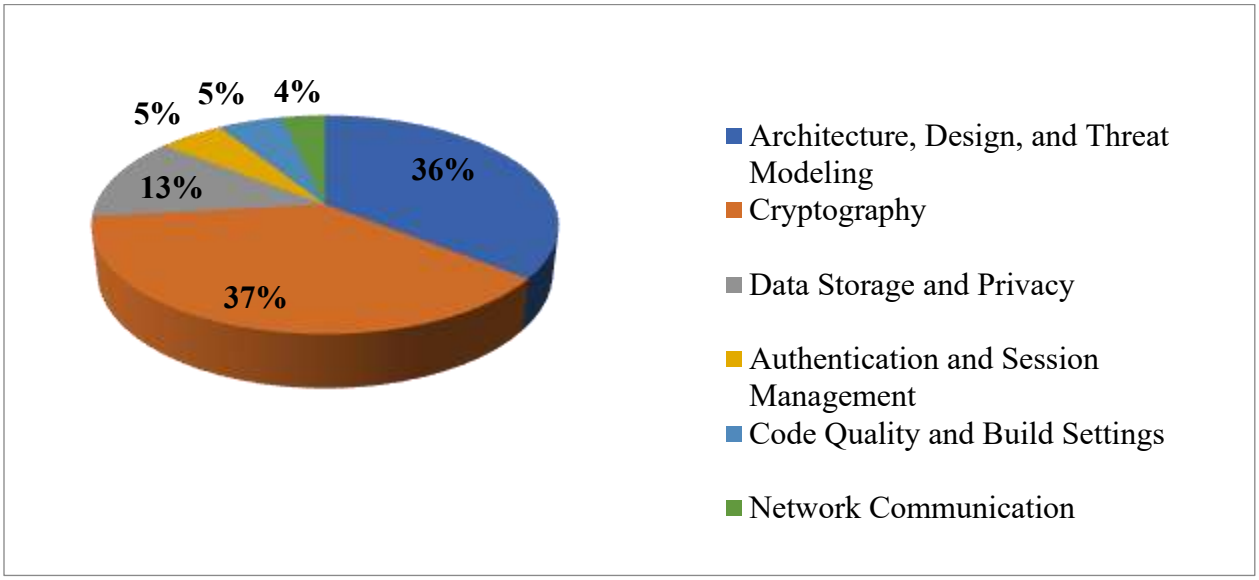


Figure 10. Categories of high-risk vulnerabilities as per MASVS framework for android applications.

protection levels of broadcast receivers, services, activities, and content providers. Furthermore, other vulnerabilities include execution of user-controlled code in WebView, and user-controlled URLs reaching WebView. Medium-risk vulnerabilities belonging to data storage and privacy category are: applications being able to read or write to external storage, hardcoding sensitive information such as usernames, keys and password into files, applications creating temporary files when sensitive financial information should never be written into temporary file, allowing application data to be backed up by any application, and allowing other applications to read intents sent for activities by setting the TaskAffinity for particular activity. Code quality and build setting category encompasses applications using SQLite databases to execute raw SQL queries, applications being vulnerable to Janus vulnerability, and IP address

disclosure. Medium risk vulnerabilities related to the cryptography category are the use of insecure random number generators, the use of weak hashing algorithms (SHA-1, MD5) known to have hash collisions, and the use of certificate algorithms vulnerable to hash collisions. The vulnerability associated with the application requesting root (super user) privileges is categorised as a resilience requirement. Being able to install the application on vulnerable Android version 8.0 or 8.1, and setting intent priority higher than in another intent are classified as being related to architecture, design and threat modelling security requirements category.

Results show that iOS applications have a higher number of high-risk vulnerabilities compared with Android applications. This could be a result of developers' assumptions about the iOS platform, believing it is secure by default, leading to less rigorous attention to security details. Also, the complexity of some of the iOS APIs is leading to improper usage, exposing the applications to high-risk vulnerabilities as observed by [49, 50]. Similar results were obtained in work by Kataria et al. [51], where it was observed that selected versions of iOS were more vulnerable than those of Android. Although the work compared platform vulnerabilities instead of applications, it still helps to dispel the notion that the iOS platform and iOS applications are secure by default.

For medium-risk vulnerabilities, Android applications tend to have more vulnerabilities than iOS applications. In general, Android applications have a larger variety of both medium and high-risk types of vulnerabilities than those observed on iOS applications. The high variety in types of vulnerabilities can be attributed to the fact that Android employs an open-source model, with lots of third-party libraries, which tend to introduce privacy leaks and increase attack surfaces [52]. However, the average number of vulnerabilities per application observed in this work is smaller than the one reported (60 per application) by Gao et al. [53] and Titiakarawongse et al. [37].

The number and types of vulnerabilities on Android applications could have been lower than they are, if applications were not allowed to install and run on Android versions that are no longer supported and have known, unpatched vulnerabilities. This pattern has not been observed with iOS-based mobile banking applications. It appears developers of mobile banking applications are placing more emphasis on accommodating as many bank customers as possible, even those who are using low-end Android Smartphones. As observed by Ndibwile et al. [54], consumers in developing countries are sensitive to the prices of smartphones. Hence, they tend to settle for low-end Android smartphones, which are resource-constrained, running unsupported Android versions (all versions before Android 9), and therefore have limited capability to handle the latest Android versions, which tend to be resource-intensive.

Other vulnerabilities related to cryptography and code quality, and build settings cut across both platforms and may be attributed to gaps in cybersecurity awareness, inadequate dependence management, and secure coding practices. This is confirmed by the significant variation in bank applications in the treatment of vulnerabilities. This is because the same tools used to develop the applications have features and functions to support secure coding techniques, such as the generation of secure random numbers and improved hashing and encryption algorithms, but these tools were not applied. These results align with those of work by [55] who studied students' approach to secure programming practices, such as avoiding the use of unsafe functions, and tested basic security knowledge, such as writing a code that always assumes user input can be malicious. The results indicated students lack the fundamental skills and

knowledge to write secure programs. This was attributed to the fact that universities' security courses are only in advanced curricula, meaning that students can complete their studies without studying them. In this work, we have observed a wide variation in the handling of vulnerabilities by various banks, such that some do not have any high-risk vulnerabilities, while others have up to six. Furthermore, the observed types of vulnerabilities indicate that programmers are using unsafe functions such as insecure random number generators, while the same frameworks they use support the generation of secure random number generators. Considerable variation of high-risk vulnerabilities among banks and observed types of vulnerabilities suggests there are knowledge, skills and experience gaps in secure programming in the banking industry.

5. Conclusion

Mobile banking enhances accessibility of financial services, reduces costs associated with paper-based statements and in-person visits, and saves time as a customer does not have to queue up to wait for a particular service. However, unauthorised access to financial information by exploiting vulnerabilities inherent in mobile banking applications may lead to financial fraud. This work sought to compare vulnerabilities based on Android and iOS mobile banking applications in Tanzania, to establish the extent to which customers' information is protected. This study has observed that mobile banking applications exhibit broad knowledge, skills, and experience gaps among banks in handling vulnerabilities. Moreover, it has been observed that iOS applications have, on average, more high-risk vulnerabilities than their Android counterparts. On the other hand, Android applications have shown a larger variety of high-risk and medium-risk vulnerabilities than iOS applications. Unlike iOS applications, some of the Android applications are developed work on vulnerable, unpatched, and unsupported Android versions.

Furthermore, while both platforms exhibit vulnerabilities related to cryptography, code quality and build settings, and network communication, Android applications additionally show vulnerabilities encompassing categories such as Platform interactions, Data Storage and Privacy, Architecture, Resilience, Design, and threat modelling. The observed vulnerabilities put sensitive customer information at risk, necessitating regulatory authorities to restrict the importation of mobile devices running platforms no longer supported. Furthermore, checks have to be done to ensure banks are adequately complying with personal data protection regulations. With regard to training of mobile application developers, more emphasis has to be paid on cryptography, code quality and build settings, platform interactions, architecture, design, and threat modeling. Future works on dynamic analysis of mobile banking applications are needed to determine privacy practices, attitudes and perceptions of developers towards security of mobile applications, commonly used versions of Smartphone platforms in Tanzania, the extent to which mobile code generators and converters mask vulnerabilities of the converted codebase, and comparison on how vulnerabilities in the banking sector fare compared with other sectors.

ACKNOWLEDGEMENT

The lab environment for this study was set by Mr. Daniel Shayo, a Mobile Application Developer from the National Environment Management Council (NEMC) of Tanzania.

CONTRIBUTIONS OF CO-AUTHORS

Daniel Ngondya	[ORCID: 0000-0003-4267-6351]	Conceived the idea, designed the experiments, conducted data collection and analysis, and wrote the paper

APPENDICES

Appendix A: Table A1. MASVS Security Requirements Categories [32].

Category of Security Requirements	**Description**
Architecture, Design, and Threat Modelling	Development applications with secure architecture which can withstand common attacks and vulnerabilities.
Data Storage and Privacy	Secure handling of data by the application to protect against leakage of Personally Identifiable Information (PII).
Cryptography	Use of strong algorithms for cryptography to protect data in transit and at rest. It includes using secure random number generators, secure protocols, and key storage management.
Network Communication	Offer protection against eavesdropping, tampering and related network attacks as traffic passes from the mobile device to remote servers.
Platform Interactions	Practices and guidelines regarding how the application interacts with different kinds of operating systems, devices, and other applications.
Code Quality and Build Settings	Codebase maintainability and secure practices in terms of configurations and settings that must be in place to build and deploy an application.
Resilience	Sustainability of service even in the face of unexpected events or tampering attempts.
Authentication and Session Management	Specifies basic requirements regarding how user accounts and sessions for remote services are managed.

Appendix B: Table A2. MASVS-CRYPTO Testing Cases[31]

SN	**Test Case ID**	**Test Case Name**
1	MASTG-TEST-0015	Testing Random Number Generation
2	MASTG-TEST-0204	Insecure Random API Usage
3	MASTG-TEST-0208	Insufficient Key Sizes
4	MASTG-TEST-0212	Use of Hardcoded Cryptographic Keys in Code
5	MASTG-TEST-0221	Broken Symmetric Encryption Algorithms

Appendix C: Table A3. List of Commercial Banks in Tanzania Offering Mobile Banking Service and Associated Names and Versions of the Mobile Banking Applications for Android and iOS.

Bank Name	Android Mobile Banking Application		iOS Mobile Banking Application	
	Name	Version	Name	Version
Amana Bank Limited	Amana Mobile	2.8	Amana Bank App	2.0
Mwalimu Commercial Bank PLC	Mwalimu Mobile	1.0.5	Mwalimu Mobile	1.0.6
Guaranty Trust Bank Tanzania Limited	GTWorld	2.2.4	GTWorld	2.2.2
China Dasheng Bank Limited	Dasheng Bank Mobile	1.0.4	Chinadashengapplicatio n	1.0.0
DCB Commercial Bank PLC	DCB Pesa	1.3.3	DCB Pesa	3.0.1
Mwanga Hakika Bank LTD	Mwanga Hakika Mobile	1.0.7	MHB Mobile Banking	1.0.9
Azania Bank Public Limited Company	Azania Bank/Azania Mobile App	2.0.7	Azania Bank	2.3
ABSA Bank Tanzania Limited	Absa Tanzania	8.3.2	Absa Tanzania	8.3.3
I & M Bank Tanzania Limited	InMTZRetail	1.0.7	I&M Bank TZ	1.2
People's Bank of Zanzibar Limited	PBZ App	1.0.1	PBZ	1.0.1
CRDB Bank PLC	Simbanking	14.0.65	SimBanking	14.0.70
Tanzania Commercial Bank PLC	TCB Popote	1.4.0	TCB Popote	1.4.7
Diamond Trust Bank Tanzania Public Limited Company	DTB m24/7	4.2.0	DTB m24/7	4.1.0
Standard Chartered Bank Tanzania LTD	SC Mobile	6.1.3	SC Mobile	6.1.3
Exim Bank Tanzania Limited	Exim Bank	5.8	Exim Online Banking-Personal	6.2

Appendix D: Table A4. List of Commercial Banks in Tanzania Offering Mobile Banking Service and Associated Names and Versions of the Mobile Banking Applications for Android and iOS.

Bank Name	Android Mobile Banking Application		iOS Mobile Banking Application	
	Name	Version	Name	Version
United Bank of Africa Tanzania Limited	UBA Mobile	8.0.84	UBA Mobile	8.0.84
Ecobank Tanzania Limited	Ecobank Mobile	4.3.5	Ecobank	4.4
Mkombozi Commercial Bank PLC	Mkombozi Mobile	1.0.2	Mkombozi Mobile	1.0.2
Canara Bank Tanzania Limited	CANMOB	3.0.8	CANMOB	3.0.8
Bank of Baroda Tanzania Limited	Baroda Tanzania Mobile	3.0.7	Bank of Baroda Tanzania	3.0.7
Stanbic Bank Tanzania LTD	Standard/Stanbic Bank	3.61.2	Standard/Stanbic Bank	3.65
Bank of Africa Tanzania Limited	B-Mobile	1.2	B-Mobile	1.9
KCB Bank Tanzania	KCB App	7.1.9	KCB	19
Akiba Commercial Bank PLC	ACB Mobile	1.0.2	ACB Mobile	1.0.8
NCBA Bank Tanzania Limited	NCBA NOW	1.0.84	NCBA NOW	1.0.84
NMB Bank PLC	NMB Mkononi	4.4.7	NMB Mkononi	5.5.47
National Bank of Commerce Limited	NBC Kiganjani	6.5.7	NBC Kiganjani	6.5.9
International Commercial Bank Tanzania LTD	ICB Digital	1.0.0	ICB Digital	20.1.14

SUPPLEMENTARY INFORMATION

Data collected for this work, particularly APK and iOS files, may be shared upon request via the author's email address: daniel.ngondya@udom.ac.tz

REFERENCES

[1]C. J. Candussi, R. Kabir, and M. Sivasubramanian, "Problematic smartphone usage, prevalence and patterns among university students: A systematic review," Journal of Affective Disorders Reports, pp. 1-12, 2023. https://doi.org/10.1016/j.jadr.2023.100643

[2]W. Malima, "Tanzania's smartphone usage surges to 25.4 million," in Daily News, ed: Tanzania Standard Newspapers Limited, 2025. [Online]. Available: https://dailynews.co.tz/tanzanias-smartphone-usage-surges-to-25-4-million/#:~:text=According%20to%20the%20latest%20report,recorded%20in%20September%20last%20year.

[3]K. P. Saidi, J. A. Jumanne, N. M. Makwinya, and N. N. Lyimo, "Determinants of Mobile Devices Adoption amongst Students: A Case of Sokoine University of Agriculture, Tanzania," East African Journal of Education and Social Sciences, vol. **6**, pp. 65-77, 2025.

[4]TCRA, "Communications Statistics Report Quarter ending December 2024," TCRA2024.[Online]. Available: https://www.tcra.go.tz/uploads/text-editor/files/Communication%20Statistics%20Report%20-%20December%202024_1736975031.pdf

[5]F. F. de Oliveira Malaquias and R. J. da Silva Júnior, "The use of m-government applications: empirical evidence from the smartest cities of Brazil," Information Technology & People, vol. 34, pp. 1357-1369, 2021. https://doi.org/10.1108/ITP-05-2020-0346

[6]K. Mouratidis, S. Peters, and B. van Wee, "Transportation technologies, sharing economy, and teleactivities: Implications for built environment and travel," Transportation Research Part D: Transport and Environment, vol. **92**, p. 102716, 2021. https://doi.org/10.1016/j.trd.2021.102716

[7]I. Pentina, L. Zhang, H. Bata, and Y. Chen, "Exploring privacy paradox in information-sensitive mobile app adoption: A cross-cultural comparison," Computers in Human Behavior, vol. 65, pp. 409-419, 2016. https://doi.org/10.1016/j.chb.2016.09.005

[8]A. Balapour, H. R. Nikkhah, and R. Sabherwal, "Mobile application security: Role of perceived privacy as the predictor of security perceptions," International Journal of Information Management, vol. **52**, p. 102063, 2020. https://doi.org/10.1016/j.ijinfomgt.2019.102063

[9]S. Tong, X. Luo, and B. Xu, "Personalized mobile marketing strategies," Journal of the Academy of Marketing Science, vol. 48, pp. 64-78, 2020. https://doi.org/10.1007/s11747-019-00693-3

[10] S. Garg and N. Baliyan, "Comparative analysis of Android and iOS from security viewpoint," Computer Science Review, vol. **40**, p. 100372, 2021. https://doi.org/10.1016/j.cosrev.2021.100372

[11] R. Gyorödi, D. Zmaranda, V. G. Adrian, and C. Gyorödi, "A comparative study between applications developed for Android and iOS," International Journal of Advanced Computer Science and Applications, vol. **8**, 2017. http://dx.doi.org/10.14569/IJACSA.2017.081123

[12] M. Alenezi and I. Almomani, "Abusing android permissions: A security perspective," in 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017, pp. 1-6.

[13]I. Mustapha, Y. Vaicondam, A. Jahanzeb, B. A. Usmanovich, and S. H. B. Yusof, "Cybersecurity Challenges and Solutions in the Fintech Mobile App Ecosystem," International Journal of Interactive Mobile Technologies, vol. 17, 2023. https://doi.org/10.3991/ijim.v17i22.45261

[14]S. Bojjagani and V. Sastry, "VAPTAi: a threat model for vulnerability assessment and penetration testing of android and iOS mobile banking apps," in 2017 IEEE 3rd international conference on collaboration and internet computing (CIC), 2017, pp. 77-86. https://doi.org/10.1109/CIC.2017.00022

[15]M. H. Juma, J. Suhonen, N. R. Mramba, M. Tedre, and A. F. Kapinga, "Smartphone Use in Financial Management among Women's Informal Saving Groups in Dodoma, Tanzania," The African Journal of Information Systems, vol. **15**, p. 2, 2023.

[16]B. Masamila, "State of mobile banking in Tanzania and security issues," International Journal of Network Security & Its Applications, vol. **6**, p. 53, 2014.

[17]N. Mori and R. Mlambiti, "Determinants of customers' adoption of mobile banking in Tanzania: Further evidence from a diffusion of innovation theory," Journal of Entrepreneurship, Management and Innovation, vol. 16, pp. 203-230, 2020. https://doi.org/10.7341/20201627

[18]G. Lamtey, "How mobile, bank customers lost billions to fraudsters," in The Citizen, ed. Dar es Salaam, Tanzania: Mwananchi Communications Limited, 2025. [Online] Available: https://www.thecitizen.co.tz/tanzania/news/national/how-mobile-bank-customers-lost-billions-to-fraudsters-5022110

[19]GBO. (2024, 16 November, 2025). Sharp rise in digital theft: Meet the crisis haunting Tanzania banking sector (B ed.). [Online] Available: https://globalbusinessoutlook.com/banking-and-finance/sharp-rise-digital-theft-meet-crisis-haunting-tanzania-banking-sector/

[20]S.-T. Sun, A. Cuadros, and K. Beznosov, "Android rooting: Methods, detection, and evasion," in Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, 2015, pp. 3-14. https://doi.org/10.1145/2808117.2808126

[21]T. H. Jackson, "Can Jailbreaking Put You in Jail, Broke," Fla. L. Rev., vol. 68, p. 631, 2016.

[22]L. Backman, "Why is security still an issue?: A study comparing developers' software security awareness to existing vulnerabilities in software applications," ed, 2018.

[23]S. Kalhoro, M. Rehman, V. Ponnusamy, and F. B. Shaikh, "Extracting key factors of cyber hygiene behaviour among software engineers: A systematic literature review," IEEE Access, vol. 9, pp. 99339-99363, 2021. https://doi.org/10.1109/ACCESS.2021.3097144

[24]D. F. Priambodo, G. S. Ajie, H. A. Rahman, A. C. F. Nugraha, A. Rachmawati, and M. R. Avianti, "Mobile health application security assesment based on owasp top 10 mobile vulnerabilities," in 2022 International Conference on Information Technology Systems and Innovation (ICITSI), 2022, pp. 25-29. https://doi.org/10.1109/ICITSI56531.2022.9970949

[25]A. Alanda, D. Satria, H. Mooduto, and B. Kurniawan, "Mobile application security penetration testing based on OWASP," in IOP Conference Series: Materials Science and Engineering, 2020, p. 012036.

[26]S. Acharya, B. Ehrenreich, and J. Marciniak, "OWASP inspired mobile security," in 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2015, pp. 782-784. https://doi.org/10.1109/BIBM.2015.7359786

[27]OWASP. (2024). Mobile Top 10 2024: Final Release Updates. [Online] Available: https://owasp.org/www-project-mobile-top-10/

[28]S. Y. Mahmud, A. Acharya, B. Andow, W. Enck, and B. Reaves, "Cardpliance:{PCI}{DSS} Compliance of Android Applications," in 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 1517-1533.

[29]S. H. Mickey and M. N. Yanhaona, "An investigation of the Online Payment and Banking System Apps in Bangladesh," arXiv preprint arXiv:2407.07766, 2024. https://doi.org/10.48550/arXiv.2407.07766

[30]O. Mykhaylova, T. Fedynyshyn, A. Datsiuk, B. Fihol, and H. Hulak, "Mobile application as a critical infrastructure cyberattack surface," Cybersecurity Providing in Information and Telecommunication Systems II 2023, vol. **3050**, pp. 29-43, 2023.

[31]OWASP. (2025, 04 August, 2025). OWASP MASTG. [Online] Available: https://mas.owasp.org/MASTG/0x05b-Android-Security-Testing/

[32]T. H. Chiboora, L. Chacha, T. Byagutangaza, and A. Gueye, "Evaluating Mobile Banking Application Security Posture Using the OWASP's MASVS Framework," in Proceedings of the 6th ACM SIGCAS/SIGCHI Conference on Computing and Sustainable Societies, 2023, pp. 99-106. https://doi.org/10.1145/3588001.3609367

[33]S. Y. Mahmud, K. V. English, S. Thorn, W. Enck, A. Oest, and M. Saad, "Analysis of Payment Service Provider SDKs in Android," in Proceedings of the 38th Annual Computer Security Applications Conference, 2022, pp. 576-590. https://doi.org/10.1145/3564625.3564641

[34]C. Anwar, S. Hady, N. Rahayu, and K. Kraugusteeliana, "The Application of Mobile Security Framework (MOBSF) and Mobile Application Security Testing Guide to Ensure the Security in Mobile Commerce Applications," Jurnal Sistim Informasi dan Teknologi, pp. 97-102, 2023. https://doi.org/10.37034/jsisfotek.v5i2.231

[35]P. V. Falade and G. B. Ogundele, "Vulnerability analysis of digital banks' mobile applications," arXiv preprint arXiv:2302.07586, 2023. https://doi.org/10.48550/arXiv.2302.07586

[36]S. Bojjagani and V. Sastry, "Stamba: Security testing for Android mobile banking apps," in Advances in Signal Processing and Intelligent Recognition Systems: Proceedings of Second International Symposium on Signal Processing and Intelligent Recognition Systems (SIRS-2015) December 16-19, 2015, Trivandrum, India, 2016, pp. 671-683. https://doi.org/10.1007/978-3-319-28658-7_57

[37]C. Titiakarawongse, S. Taksin, J. Ruangsawat, K. Deeduangpan, and S. Boonkrong, "Comparative Vulnerability Analysis of Thai and Non-Thai Mobile Banking Applications," Journal of Cybersecurity and Privacy, vol. 4, pp. 650-662, 2024. https://doi.org/10.3390/jcp4030031

[38]P. Thomson, "Static Analysis: An Introduction: The fundamental challenge of software engineering is one of complexity," Queue, vol. **19**, pp. 29-41, 2021.

[39]R. Sihwail, K. Omar, and K. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," Int. J. Adv. Sci. Eng. Inf. Technol, vol. 8, pp. 1662-1671, 2018. https://doi.org/10.18517/ijaseit.8.4-2.6827

[40]W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," IEEE transactions on software engineering, vol. **43**, pp. 817-847, 2016. https://doi.org/10.1109/TSE.2016.2630689

[41]BoT. (2024, 01 August, 2024). List of Licensed Institutions. [Online] Available: https://www.bot.go.tz/BankSupervision/Institutions

[42]D. Lakens, "Sample size justification," Collabra: psychology, vol. **8**, p. 33267, 2022.
[43]A. S. Singh and M. B. Masuku, "Sampling techniques & determination of sample size in applied statistics research: An overview," International Journal of economics, commerce and management, vol. **2**, pp. 1-22, 2014.

[44]A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in Proceedings of the 15th international conference on mining software repositories, 2018, pp. 181-191. https://doi.org/10.1145/3196398.3196401

[45]K. Kandasamy, S. Srinivas, K. Achuthan, and V. P. Rangan, "IoT cyber risk: A holistic analysis of cyber risk assessment frameworks, risk vectors, and risk ranking process," EURASIP Journal on Information Security, vol. 2020, p. 8, 2020. https://doi.org/10.1186/s13635-020-00111-0

[46]D. Bassolé, G. Koala, Y. Traoré, and O. Sié, "Vulnerability analysis in mobile banking and payment applications on android in African Countries," in Innovations and Interdisciplinary Solutions for Underserved Areas: 4th EAI International Conference, InterSol 2020, Nairobi, Kenya, March 8-9, 2020, Proceedings 4, 2020, pp. 164-175. https://doi.org/10.1007/978-3-030-51051-0_12

[47]P. Techapalokul and E. Tilevich, "Novice programmers and software quality: Trends and implications," in 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T), 2017, pp. 246-250. https://doi.org/10.1109/CSEET.2017.47

[48]S. S. El-Kaliouby, A. H. Yousef, and S. Selim, "Mobile Application Code Generation Approaches: A Survey," in International Conference on Model and Data Engineering, 2022, pp. 136-148. https://doi.org/10.1007/978-3-031-23119-3_10

[49]Y. Li, Y. Zhang, J. Li, and D. Gu, "icryptotracer: Dynamic analysis on misuse of cryptography functions in ios applications," in International Conference on Network and System Security, 2015, pp. 349-362.

[50]Z. Mousavi, C. Islam, M. A. Babar, A. Abuadbba, and K. Moore, "Detecting misuse of security APIs: A systematic review," ACM Computing Surveys, vol. **57**, pp. 1-39, 2025. https://doi.org/10.1145/3735968

[51]A. Kataria, T. Anjali, and R. Venkat, "Quantifying smartphone vulnerabilities," in 2014 International Conference on Signal Processing and Integrated Networks (SPIN), 2014, pp. 645-649. https://doi.org/10.1109/SPIN.2014.6777033

[52]X. Zhan, T. Liu, L. Fan, L. Li, S. Chen, X. Luo, et al., "Research on third-party libraries in android apps: A taxonomy and systematic literature review," IEEE Transactions on Software Engineering, vol. 48, pp. 4181-4213, 2021. https://doi.org/10.1109/TSE.2021.3114381

[53]J. Gao, L. Li, P. Kong, T. F. Bissyandé, and J. Klein, "Understanding the evolution of android app vulnerabilities," IEEE Transactions on Reliability, vol. **70**, pp. 212-230, 2019. https://doi.org/10.1109/TR.2019.2956690

[54]J. D. Ndibwile, E. T. Luhanga, D. Fall, D. Miyamoto, and Y. Kadobayashi, "Smart4gap: Factors that influence smartphone security decisions in developing and developed countries," in Proceedings of the 2018 10th International Conference on Information Management and Engineering, 2018, pp. 5-15. https://doi.org/10.1145/3285957.3285980

[55]J. Lam, E. Fang, M. Almansoori, R. Chatterjee, and A. G. Soosai Raj, "Identifying gaps in the secure programming knowledge and skills of students," in Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume **1**, 2022, pp. 703-709. https://doi.org/10.1145/3478431.3499391