**International Journal of Scientific Research in Multidisciplinary Studies**
Vol.**10**, Issue.**7**, pp.**29-45**, July **2024**
E-ISSN: 2454-9312 P-ISSN: 2454-6143
Available online at: www.isroset.org

Research Article

# Assessing the Vulnerabilities of Online Digital Mobile Banking Applications in Nigeria

**Grace Bunmi Akintola**[1*]

[1]Dept. of Cyber Security, Nigerian Defence Academy, Kaduna, Nigeria

**Corresponding Author: gbogundele@nda.edu.ng*

**Received**: 10/May/2024; **Accepted**: 11/Jun/2024; **Published**: 31/Jul/2024

***Abstract***— In this digitalized era, people prefer conducting their banking transactions and services remotely, anywhere, any day, and anytime without visiting their local banks physically. This has caused a great increase in the utilization of mobile devices by users such as Android phones on which mobile bank applications are being installed to carry out financial transactions. Despite the numerous benefits of using various mobile banking applications, there are huge issues concerning security and privacy aspects that can be leveraged by malicious hackers such as task hijacking which involves taking over the legitimate user's activity to perform further cyber attacks, thereby, causing harm (such as Denial of Service attacks) to the users. Therefore, this paper focuses on assessing the vulnerabilities of twelve (12) selected Nigerian digital banks' mobile applications installed on an Android mobile device using the Ostorlab which is a mobile security testing tool. The selected digital banks' mobile apps include Kuda, moniepoint, fundall, carbon, sparkle, vbank, palmpay, opay, mintyn, eyowo, ALAT and Fairmoney banks. A total number of 32 vulnerabilities were discovered generally while sixteen (16) vulnerabilities were most commonly found across all the mobile applications. Vbank was discovered to have the maximum number of vulnerabilities with 68.75% while Fundall bank had the least vulnerabilities with 46.88%.

***Keywords***— banking, mobile bank application, security, privacy, vulnerabilities, digital banks, Ostorlab

## 1. Introduction

Banking is described as an industry that controls credit facilities, cash storage, investments, and various financial transactions. As a key driver of many economies, it helps in conveying funds to borrowers with productive investments. With the appearance of digital technologies, the financial industry has undergone a dramatic change over the last few years [1]. On the high demand of customers for a more customized, flexible, and smooth-running banking experience, digitalization has become essential for banks to remain competitive. This evolution has proceeded with the growth of new products and services, including mobile banking, online payments, and robot-advisory [2].

The inception of digital banking has revolutionized the financial services industry, reshaping how banks interact with customers and conduct business. This transformation is not merely technological but a fundamental change in the banking paradigm, affecting customer behavior, business models, and regulatory landscapes. Digital banking's significance lies in its ability to enhance accessibility, efficiency, and innovation in financial services, making it a critical area of study in contemporary banking and finance [3].

In Nigeria, digital financial services have profoundly influenced the economy. Research shows that services such as mobile banking and point-of-sale transactions have positively impacted Nigeria's Gross Domestic Product (GDP), highlighting the economic significance of digital banking innovations. This impact is especially noteworthy in developing economies, where digital banking can catalyze economic growth and financial inclusion [4]. Over the past decade, the digital revolution has significantly impacted banking. Digital banking platforms, including mobile applications and websites, have proliferated, offering consumers and banks an alternative to traditional brick-and-mortar branch networks. This technology has emerged as the primary means through which consumers access financial services and banks deliver them. [5].

The numerous benefits of adopting digital banks have greatly influenced the growing number of users of digital banks' mobile applications for their day-to-day financial transactions at any time, any day, and anywhere. However, despite the digital banks' mobile application usefulness, there's an increasing number of security vulnerabilities present in the digital banks' mobile apps which can be easily exploited by a malicious hacker to inappropriately get hold of legitimate users' sensitive information, thus, using them to launch further

 

cyber-attacks. Therefore, this paper aims to assess the vulnerability of twelve selected online digital mobile banking applications (carbon, sparkle, ALAT, FairMoney, Fundall, palmpay, Opay, Kuda, Vbank, moniepoint, eyowo, and Mintyn banks) in Nigeria using the Ostorlab security testing tool in which the obtained results were analyzed to identify the bank with the highest vulnerabilities to the one with the least vulnerabilities as well as countermeasures to be applied and suggesting further works in using other security testing tools to generate better results.

This research paper is organized into several sections. Section 1 provides an introduction to banking and digital banking. Section 2 covers related work, including an overview of digital banks, mobile devices, and applications, and various techniques for analyzing digital banks' mobile apps. Section 3 details the research methodology, explaining the procedures involved in adopting the Ostorlab tool to analyze the mobile applications of twelve selected Nigerian digital banks. Section 4 discusses the generated results from the analyzed mobile application files, highlighting identified vulnerabilities and countermeasures. Lastly, Section 6 concludes the research with future recommendations.

## 2. Related Work

### 2.1 Overview of digital banks

The world has become digitalized, specifically in the banking sector. It has gradually moved from traditional banking to digital banking as it makes it easier for people to make all transactions with their mobile devices instead of the physical presence of users at the local banks [6]. Another benefit of adopting digital banking was the comfort it gave in performing financial transactions at any time [7]. Digital banking also helps to elevate the banking sectors' efficiency and profitability as well as increase the 4Cs which means Cost, Convenience, Control, and Customer satisfaction which was its primary goal. Governments are advocating for cashless, paperless, and faceless transactions in the economy. Digital payment systems entail key stages: signing up, billing, selecting a payment method, and confirming payments. The term "digital" refers to data storage as digital signals, simplifying banking operations and streamlining transactions. SMS banking serves as an illustration. Consumers can now complete tasks at their convenience [8]. It was clearly stated by [9] several benefits of digital banks from both customer and business perspectives. Advantages of the usage of digital banks from the customer's view include bringing convenience, improving security, and providing self-services while advantages from the business view include reducing the costs of transactions when compared to traditional banks, ability to easily adapt to changes in the market place, ability to increase the number of customers through use of bank's digital channels and improving customer experience.

Regardless of the innumerable benefits of the usage of digital banks, there are also disadvantages associated with it [8] and these include security and privacy issues since there are not 100% safe transactions, thus, making some bank users' views about digital banking remain unchanged. Another challenge was the uninformed knowledge of the users about the implementation and periodical updates of the digital banking system for improved security of the customer's transactions which may sometimes cause sudden disruption of transaction activity intended to be carried out by customers at a particular time without warning them. [10]. Another drawback was the internal barriers as it relates to both bank customers' and personnel's level of knowledge about digital transactions in determining how the expansion of the digital banking system. Finally, these days, the demand for digitization is rapidly increasing. However, most banks lack the courage to implement this strategy immediately.

### 2.2 Mobile devices and applications

According to [11] mobile device can be simply defined as a portable computing device, characterized by its small form factor for easy individual carrying, operates wirelessly, has local data storage, and features a self-contained power source. Examples of mobile devices include phones and PDAs (Personal Digital Assistants) which offer a more affordable means of accessing the internet anytime, anywhere in performing various activities despite the higher connection costs compared to desktop computers. In this era, most activities are being carried out by the use of mobile phones such as banking transactions as they are supported by mobile operating systems which are software that enable smartphones, tablets, and other devices to execute applications and programs [12]. Today, there are different mobile operating systems available, with two of the most popular being Apple's iOS, utilized by iPhones, and Google's open-source Android operating system and they allow several mobile applications to be installed easily on the devices as the user's desire. Mobile application is clearly described as a form of application software specifically crafted to work on mobile devices like smartphones or tablet computers and examples of mobile applications include mobile bank apps. Several benefits of using mobile applications include ease of use, accessibility, user-friendly, consumption of lesser of time, and convenience. However, there are challenges to using mobile applications such as Privacy and data concerns, difficult for users who are technologically inexperienced to understand which applications to install and use [13]. Digital bank mobile application has been one of the most utilized apps by users across the world as it made it easier to install mobile apps on mobile phones to perform any form of transaction and banking services online such as creating a bank account and making payments.

### 2.3 Various techniques for analyzing digital banks' mobile apps

Analyzing the loophole of six different UK digital banks' mobile applications was focused on using the Androbug tool, a vulnerability scanner. Among the identified vulnerabilities, Revolut's mobile application had the maximum number, whereas Starling's had the fewest [14].

A reverse engineering method was adopted in analyzing mobile banking applications using available tools. A dataset of mobile banking applications that offer banking services in Pakistan was collected for the research. The results indicated

 

that none of the current tools can implement the entire reverse engineering process independently. Furthermore, notable variations in both the time of implementation and the quantity of files created by each tool when applied to the same file were observed [15].

Another research was conducted which aimed to analyze the prevalent usage of digital banking services among bank customers by employing text mining techniques on a dataset comprising their feedback. The study encompassed digital banking data from the ten most utilized private banks and three state-owned banks, totaling thirteen banks, based on data from the Banks Association of Turkey spanning from January 2020 to August 2022. Approximately 1,200,000-1,250,000 raw data entries were derived from various social media platforms where customers discussed these banks. Each bank was individually scrutinized, applying analyses on word density, creating word cloud visualizations, and conducting sentiment analyses to gauge customer perspectives. The study revealed that customers most frequently mentioned aspects such as the ease of use, utility, and service fees of digital applications. Consequently, it was deduced that while there's no consequential difference between private and public banks in terms of digital services, private banks excel in terms of utility and innovation. Based on the analysis, recommendations were formulated for banks to enhance customer satisfaction and improve the quality of service delivery in digital banking [16].

A research was created to textual data extracted from 37,460 reviews by users of Nigerian mobile banking applications ranging from November 2012 to July 2020. Out of a rating of 5, the twenty-two (22) apps in the sample had a 3.5 user rating. Non-interest bank apps had the maximum average rating (4.0), while commercial banks with national authorization had the minimum (3.4). Sentiment analysis shows that positive sentiment words make up 17.8% of the corpus, more than double the 7.7% of negative sentiment words. About 66% of the emotions expressed by users include 'confidence,' 'eagerness,' and 'joy,' while the rest relate to 'shock,' 'anxiety,' 'rage,' and 'detestation.' This suggests that most users are contended with their mobile banking experience. The main topics in the reviews include feedback on banks' responsiveness to user complaints, user experience with app functionalities and updates, and operational failures with the apps. The results emphasize the importance of banks promoting awareness of app functionalities, enlightening users on accessing these features, and acknowledging reports on time and accurately [17].

The research conducted deals with investigating the adoption of mobile banking technology by consumers by examining various factors such as effective commitment, easy transaction, ease to use, perceived dependability, before and after benefits, service, system, and information quality, bank trust, and profitability. The study utilizes a simple linear regression method with SPSS software to analyze the inter-relationship between these variables and the acceptance of mobile banking technology. It explores hypotheses and identifies the relationships that exist between these parameters [18]

Using the case study of Saudi Arabia, research was conducted to analyze, evaluate, and compare the functionality of all available mobile banking apps in Saudi Arabia for both iOS and Android platforms. Usability, defined by ISO 9241, was assessed based on productiveness, competency, and satisfaction criteria. The study pinpointed and explained the primary loopholes of these apps in supplying standard solutions to users. Critical issues were found in-app user interfaces and functionality, particularly in those frequently updated. Additionally, weak customer support was noted, leading to dissatisfaction among customers and undermining interaction between banks and users. [19]. The effectiveness of digital banking services offered by a bank in Malaysia was investigated, specifically focusing on the benefits, dependability of the banking system, and the effect of the COVID-19 pandemic. Utilizing the Technology Acceptance Model (TAM), data was collected through a questionnaire survey involving 228 bank clients. Results revealed that perceived usefulness and banking system reliability notably affected digital banking effectiveness. However, the COVID-19 pandemic was found to have no significant influence on the value of digital banking from the perspective of bank clients. These findings offer a perception of the bank's future banking strategies and address consumer needs, while also contributing to an understanding of the Malaysian financial industry through the lens of a prominent financial institution [20].

A study on the risks and weaknesses surrounding mobile banking apps within the Kenyan sector was conducted. It was realized that despite their convenience, these apps are susceptible to threats compromising user details and transactions. The research helps to discover these risks and loopholes peculiar to Kenyan mobile banking apps and propose solutions to bolster their security. It pursues mainly three objectives: evaluating primary vulnerabilities and attack vectors, assessing common flaws, and recommending tailored security measures. These include cryptography, two-factor authentication, and user awareness pieces of training. By conducting an in-depth analysis and offering practical advice, the study seeks to aid banking companies and customers in fortifying the security of their mobile banking experiences in Kenya [21], A security analysis of five popular e-wallets and five leading mobile banking apps in Malaysia was conducted. The analysis was based on security principles recommended by OWASP under the Mobile Security Testing Guide (MSTG) and Mobile Security Threats (MST). Static analysis was performed in which three mobile application-testing tools were utilized, comprising vulnerability scanning, code review, and penetration testing. While all apps met security requirements, their security elements, such as encryption, security protocols, and app services, varied significantly [22]. A descriptive research approach and desktop research were adopted to examine growing security threats in mobile banking applications. It recognized and assessed advanced systems to lessen these threats and highlighted key open research issues in the field [23]. Finally, research was done, focusing on m-banking applications for Android OS, specifically examining their security, vulnerabilities, threats, and potential solutions. Two mobile testing frameworks were

 

used to analyze and benchmark the applications against best practices. The study identified security weaknesses in the apps, indicating the necessity for a more thorough security evaluation in Qatar to enhance user confidence. Understanding the security opinion is crucial for improving m-banking security and user awareness [24].

## 3. Research Methodology

From a comprehensive literature review conducted, several approaches and techniques have been adopted by researchers which were used for discovering and analyzing security vulnerabilities in different mobile applications using various tools. Therefore, this section discussed the research methodology, the tool adopted, and the selected Nigerian online digital banks which were analyzed by the Osorlab security testing tool.

### 3.1 Justification of the Adopted Research Method

A quantitative research type was adopted in this paper as it involves analyzing each digital bank mobile application file (.apk) for its security vulnerabilities as well as its overall risk level, and the number of vulnerabilities found in each mobile app, the quantitative research methodology was selected due to its easy understanding of the data (results) which are represented in graphical form and also categorized based on low, medium and high level of vulnerabilities. Another advantage of using a quantitative method for the research is its high level of accuracy [25].

.

### 3.2 Justification of the selected tool

Ostorlab, one of the best mobile security testing tools that helps in effectively discovering security loopholes in Android and iOS devices, was selected for this study. Out of various mobile security testing tools available, Ostorlab was selected for this research due to the autonomous security solutions it gives. It is applicable for scanning security and privacy issues as well as the components of mobile, web, and API applications. The benefits of choosing the Ostorlab tool include its user-friendly interface which makes it easily accessible to the security tester. It offers fast and actionable insights and recommendations that can help organizations and developers quickly take necessary security measures to the identified vulnerabilities. It provides an expensive range of security testing techniques for deep scanning of mobile applications for all potential vulnerabilities, that is, conducting advanced analysis in terms of loopholes in dependencies, insecure programming patterns, privacy issues, and intercepted backend communication in identifying server-side vulnerabilities. [26]

**Step-by-step procedure for using the Ostorlab tool for the digital bank mobile application analysis**

1. Create an account with Ostorlab
2. On the dashboard page, click on the scanning menu on the left side
3. Click on the new scan
4. On the new scan page, tag the name of the mobile application at the “asset tab to be analyzed.
5. Select the type of device (Android or iOS) to retrieve the mobile application file.
6. Click and navigate to the location of the mobile application file (.apk file)
7. Select scan type to determine the type of analysis to perform
8. Click on the submit button

After properly using the Ostorlab tool to upload the mobile application, it took some time for the analysis results to be submitted to the tester’s registered email used in creating the account. The obtained results are being analyzed as all vulnerabilities identified were shown, including their risk level and recommended solutions. The step-by-step procedure was repeated for all the selected digital mobile applications to get the results.

### 3.4 The Nigerian online digital banks

Out of various Nigerian online digital bank mobile applications, twelve (12) digital banks were selected based on the comprehensive review conducted which shows the most common digital banks used by people to make transactions. As shown in Table 1, eight different authors talked about the commonly used digital banks in Nigeria. The symbol “1” indicates the presence of a particular bank mentioned by each author.

Table 1: The selected digital banks in Nigeria

| Online digital banks | [27] | [28] | [29] | [30] | [31] | [32] | [33] | [34] |
|---|---|---|---|---|---|---|---|---|
| **Kuda bank** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Opay** | 1 | | 1 | | | 1 | | 1 |
| **Sparkles** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **Mintyn** | 1 | | | 1 | 1 | 1 | 1 | |
| **Palmpay** | 1 | | 1 | | | 1 | | 1 |
| **Moniepoint** | 1 | | | | | | | 1 |
| **Fundall** | 1 | | 1 | | 1 | | | |
| **VBank** | 1 | 1 | 1 | 1 | 1 | | | 1 |
| **FairMoney** | 1 | | | 1 | | | 1 | |
| **Carbon** | 1 | | | 1 | | | | 1 |
| **ALAT** | | 1 | 1 | 1 | 1 | | | 1 |
| **Rubies** | | 1 | | 1 | 1 | | | |
| **oneBank** | | 1 | 1 | | 1 | | | |
| **Eyowo** | | 1 | | 1 | 1 | | 1 | |
| **Gomoney** | | | 1 | 1 | | | | |

 

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Piggyvest** | | | | 1 | | | 1 | |
| **Smartsaver** | | | | 1 | | | | |
| **Cowriewise** | | | | 1 | | | | |
| **Alladin** | | | | 1 | | | | |
| **SumoTrust** | | | | 1 | | | | |
| **JetSeed** | | | | | | | | |
| **Kolopay** | | | | 1 | | | | |
| **goMoney** | | | | | 1 | | | |
| **Sofri** | | | | | | | 1 | |
| **Branch App** | | | | | | | | 1 |
| **GTworld App** | | | | | | | | 1 |

Figure 1 explains the chart's diagram of the listed digital banks by various researchers as shown in Table 1. Generally, a total number of 25 digital banks were listed as shown in the chart, and based on the scale of 0 to 80, twelve (12) banks were selected because they had the highest number of scales that is, frequently mentioned by the researchers. The digital banks with the highest scale include Kuda, Opay, Sparkles, Mintyn, palmpay, moniepoint, fundall, Vbank, FairMoney, carbon, ALAT, and Eyowo banks. Kuda and sparkles were the most common banks mentioned by all the eight researchers followed by the Vbank mentioned by seven out of eight researchers followed by Mintyn and ALAT banks shown by five out of eight researchers. Eyowo and palmpay banks were mentioned by four researchers. Finally, Fundall, carbon, and FairMoney were mentioned by three researchers.

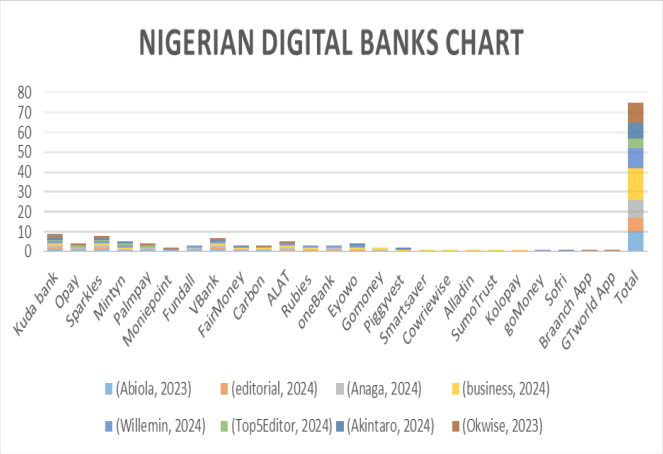

Figure 1: Nigerian Digital Banks chart

### 3.5 The current OWASP Top 10 standard for Mobile Applications

The Open Web Application Security Project (OWASP) Top 10 standard is a regularly updated report highlighting web applications' most critical security concerns. Compiled by a global team of security experts, this report acts as an 'awareness document.' OWASP proposes that all companies make use of the findings of the Top 10 in their processes to minimize and mitigate security risks [35].

Viewing particularly about mobile applications, there are the latest Top 10 OWASP Mobile standards which list common security vulnerabilities discovered in mobile applications. This can be used as a framework for assessing various vulnerabilities in mobile applications. These include:

a. Inappropriate Credential Usage: Malicious hackers can trick the loopholes in hardcoded credentials and inappropriate use of confidential information in mobile applications. Once identified, the vulnerabilities permit attackers to adopt hardcoded information to obtain illegal access to the confidential capability of the app. Additionally, they can misuse validated or stored credentials, bypassing legitimate access requirements.
b. Inadequate Supply Chain Security: Exploiting mobile app supply chain weaknesses, a malicious hacker can manipulate application functionality. For instance, malicious code may be inserted into the app's codebase or alter the code during the build process to establish backdoors, spyware, or other harmful software. This could empower the malicious hacker to steal data, surveil users, or take over the mobile device.
c. Insecure Authentication/Authorization: Once adversaries identify loopholes in authentication or techniques, they can exploit these weaknesses in two ways. They may either falsify or avoid authentication through direct submission of service requests to the mobile app's backend server, avoiding any direct communication with the mobile app. Alternatively, they can be granted access as a valid user, pass the authentication mechanism, and then forcefully view a vulnerable endpoint to carry out administrative activity.
d. Insufficient Input/Output Validation: Lack of proper input/output validation reveals the application to critical attack vectors such as SQL injection, XSS, command injection, and path traversal. These loopholes may result in illegal access, manipulating data, code execution, and compromise of the whole backend system.
e. Unprotected Communication: Most recent mobile applications make data transfer with one or more remote servers. During transmission, the data typically passes through the mobile device's carrier network and the internet. If the data is conveyed in plaintext or using a deprecated encryption protocol, a threat agent snooping on the wire can obstruct and alter it.
f. Inadequate Privacy controls: Privacy controls safeguard Personally Identifiable Information (PII) such as names, addresses, credit card details, email addresses, and IP addresses, and details about health, religion, sexuality, and political opinions. To obtain PII, attackers must first breach another layer of security. Using a trojan, they could intercept network communication, obtain access to the file system, clipboard, or logs, or physically obtain the mobile device and create a backup for analysis.

 

g. Insufficient Binary Protections: App binaries can typically be downloaded from app stores or copied from mobile devices, making binary attacks relatively easy to set up. These binaries are susceptible to two major types of attacks: reverse engineering and code tampering. Reverse engineering involves decompiling and scanning the app for valuable information, such as secret keys and vulnerabilities. Code tampering involves manipulating the app binary to include malicious code.
h. Security Misconfiguration: Security misconfigurations in mobile apps can be exploited through various attack vectors, including insecure default settings, improper access control, weak encryption, insecure communication, unprotected storage, insecure file permissions, and misconfigured session management.
i. Insecure Data Storage: Insecure data storage in a mobile application creates loopholes that threat actors can utilize various attack vectors. These vectors include an unauthorized approach to the device's file system via physical or remote means, taking advantage of weak or non-existent encryption, data transmission interruption, and using malign apps installed on the device.
j. Insufficient Cryptography: The attack vector for weak cryptography in a mobile application deals with taking advantage of vulnerabilities in the cryptographic mechanisms designed to secure confidential data. Attackers may use different methods, including cryptographic attacks, brute force attacks, or side-channel attacks, to take advantage of weaknesses in encryption algorithms, key management, or implementation flaws.

## 4. Results and Discussion

### 4.1 Description of the identified vulnerabilities in the digital bank application using the Ostorlab tool

**Insecure object Serialization:** this is an identified vulnerability in the bank application. This shows that the application employs an insecure deserialization scheme for handling untrusted data. Insecure object deserialization can lead to arbitrary remote code execution, manipulation of application logic, and data tampering, including bypassing access controls. Exploiting deserialization vulnerabilities is challenging as it requires customizing exploits for targeted attacks due to the lack of readily available exploits.

**Undeclared permissions:** This vulnerability involves applications exposing their utility to other apps by defining permissions that those apps can request. To enforce permission, it must first be declared in the AndroidManifest.xml using the <permission> element before applying it to components using "android:permission=". If the application requests permission without clearly stating it, a malicious app can declare that permission with a normal protection level, invoke it, and access the protected component of your application. Furthermore, undeclared permissions can serve as a security risk, as the authorized user may be ignorant of the app retrieving confidential information or system resources. This could proceed to a breach of privacy or any other security issues, such as an app viewing the user's location without their consent.

**Biometric Authentication bypass:** A robust implementation of mobile biometric authentication ensures that accessing the application's confidential data requires Face ID or Touch ID authentication. This secure approach extends beyond mere fingerprint or face verification for login. It involves encoding the application's confidential data using biometric data, adding a layer of protection. This encryption significantly increases the difficulty for unauthorized individuals to access or misuse sensitive information. Encrypting data with biometric authentication becomes crucial in scenarios where unauthorized parties gain device access, whether through malware or physical means. Android provides mechanisms to enforce biometric authentication to secure confidential information. Biometric authentication has evolved to provide improved user experience, developer experience, and improved security. Previous implementation using FingerprintManager is deprecated and must not be used. Proper implementation must use BiometricManager with Biometric Prompt and CryptoObject. Biometric authentication is done without CyptoObject or a secret key, making it vulnerable to biometric bypass.

**Attribute usesCleartextTraffic set:** The android:usesCleartextTraffic attribute determines whether the app plans to utilize cleartext network traffic, like cleartext HTTP. For apps targeting API level 27 or lower, the default value is "true", while for those targeting API level 28 or higher, the default is "false". The attribute is not explicitly set and Android Network security configuration is not defined, thus, allowing unauthorized users to take advantage of this loophole to violate the legitimate user's privacy and security.

**Task Hijacking:** This is a security vulnerability that allows malicious apps to take over the tasks of legitimate apps, tricking users into disclosing secret information or carrying out unwanted actions. For instance, a malicious app called Camero was found to be exploiting task hijacking to steal banking credentials. The app masqueraded as a legitimate camera app but would launch a fake banking login screen when users opened their banking apps. Task hijacking is a serious threat to Android users and businesses alike and has been identified across all the selected bank applications which can result in privacy and security issues, targeted by malware, and has publicly available exploits.

**Services declared without permissions:** A service is an application component responsible for executing actions in the background without requiring user interaction. It can also expose functionalities to other applications, typically through calls to Context.bindService() to establish a connection and interact with it. However, unprotected services can be requested by other applications, potentially resulting in illegal access to confidential information or the execution of privileged actions.

**Secret information stored in the application:** exposing secrets, passwords, and API keys can bring forth serious consequences, such as Loss of confidentiality (unauthorized individuals gaining access to sensitive information), loss of integrity (gaining unauthorized access to systems or data,

 

potentially leading to the alteration or corruption of that information), loss of availability (the unauthorized use of leaked secrets or passwords may result in the denial of access to legitimate users, leading to a loss of availability of the affected systems or data), reputational damage (suffering damage to company's reputation), legal consequences (depending on the nature of the leaked information and the laws in the relevant jurisdiction, the leakage of secrets or passwords could potentially lead to legal consequences), overbilling (Leaking API keys could potentially be used by illegal individuals to access API resources and perform actions that may incur charges, leading to overbilling).

**Use of WiFi API that contains or leaks sensitive PII:** The application utilizes the ACCESS_WIFI_STATE interface and invokes APIs like getConnectionInfo to retrieve sensitive information regarding the Wi-Fi access point, such as BSSID, SSID, and RSSI, as well as device details like MAC address and IP address. However, this API is prone to abuse, allowing access to personally identifiable information (PII) such as unique device identifiers through the device's MAC address, geolocation data via surrounding Wi-Fi access points, and tracking of user travel history and social connections by monitoring users connecting to the same access point. This can be used by the attacker to perform further attacks, violating the legitimate user's privacy and security.

**Unimplemented activity class detected:** The analysis identified an activity class declared in the manifest, such as <activity android:name='ClassName'>, but no corresponding implementation for this class was found within the application's codebase. This discrepancy could lead to runtime errors or dysfunctional behavior during the application's execution.

**Application checks rooted device**: The presence of strings and methods suggest the need to check for rooted or Jailbroken devices. The absence the Jail-broken or Root detection is not a vulnerability, but its presence remediates the impact of certain vulnerability classes or threats.

**Ddebug mode disabled**: The application is compiled with debug mode disabled. Debug mode, when enabled, permits unethical hackers to access the application's filesystem and attach a debugger to retrieve sensitive data or execute malicious actions. However, the issue is that the **debuggable** attribute is not set and its default value is false.

**Backup mode disabled /enabled**: Backup mode is a feature in Android that allows users to backup and restore data and settings from one device to Another. By default, Android conducts a complete backup of applications, encompassing private files stored on the /data partition. Subsequently, the Backup Manager service transfers this data to the user's Google Drive account. The issue is that the allowBackup attribute is set to false**.**

**Exported activities, services, and broadcast receiver's list:** The list includes all exported components within the application. Exported components are accessible to external applications and serve as entry points to the application. However, there are technical issues in the exported services, receivers, and activities definitions.

**Android Manifest:** The manifest file provides important details about your app to the Android system, which is necessary before the system can execute any of the app's code. However, there appears to be a technical issue in the AndroidManifest.xml file.

**Implementation of a WebViewClient:** it contains the List of WebviewClient implementations. The WebView class, extending Android's View class, facilitates the display of web pages within your activity layout. Essentially, you can incorporate a WebView layout in XML layout files and subsequently load a URL to be presented to the user. The issue is found in the List of webview implementations.

**Calls to native methods:** It comprises a list of all method calls utilizing the Java Native Interface (JNI) to interact with native code, typically written in C/C++.This issue was the List of detected **Java** native methods.

**Broadcast receiver's dynamic registration:** One or more broadcast receivers in the application are dynamically registered in the code and lack protection by signature permission in the AndroidManifest.xml file. Additionally, all dynamically registered receivers are exported. This vulnerability allows an attacker, using malware, to broadcast arbitrary data to the exported receiver, potentially triggering the invocation of various components of the application or even executing code.

**Insecure network configuration setting:** Android Network Security Configuration allows for a declarative setting of the application's network security. However, the application lacks a Network Security Config.

**Unimplemented service class detected:** The analysis identified a service class declared in the manifest, such as <service android:name='ClassName'>, but no corresponding implementation for this class was found within the application's codebase. This discrepancy could lead to runtime errors or dysfunctional behavior during the application's execution.

**FileObserver Implementation:** It monitors files using Inotify to trigger an event after files are accessed or modified by any process on the device, including the application itself. FileObserver is an abstract class; subclasses need to implement the event handler onEvent(int, java.lang.String). Each FileObserver instance can monitor multiple files or directories. Monitoring a directory triggers events for all files and subdirectories within it. An event mask specifies the changes or actions to report, with event-type constants describing potential changes in the event mask and detailing occurrences in event callbacks. The technical issue was that the Application was using the **FileObserver** API.

**Insecure File Provider Paths Setting:** The application utilizes androidx.core.content.FileProvider to expose a file provider. Within the provider, available files are specified in the metadata child attribute named android.

 

Support.FILE_PROVIDER_PATHS**.** The application uses a content provider xamarin.essentials.fileProvider that reveals a file provider android. Support.FILE_PROVIDER_PATHS configured in res/xml/xamarin_essentials_fileprovider_file_paths.xml fil e. The file provider has an external path type that allows the attackers to access external storage like an SD card.

**Attribute requestLegacyExternalStorage Set:** The `android:requestLegacyExternalStorage` attribute allows access to directories and various types of media files stored in external storage. However, it is effective only for Android 10 (API level 29). On Android 11 and later versions, the system disregards the `requestLegacyExternalStorage` flag.

**Deprecated API version:** The android:targetSdkVersion attribute specifies the Android Target API level required by the application. Setting a low targetSdkVersion may allow the application to run on older Android versions but could expose users to security vulnerabilities.

**OAuth Account Takeover by hijacking custom schemes:** The vulnerability arises from the application's use of a custom scheme in the redirect_uri parameter during OAuth authentication. In a typical OAuth scenario, redirect_url should be guaranteed to belong to the client application (identified by client_id) that requests data from an identity provider (Google, Facebook, Github...). Using a custom scheme breaks that premise as it can be claimed by the application on the user's device. Attackers can **bypass user interaction** by leveraging certain techniques like express authentication flow or using OAuth parameters that are meant to skip the consent prompt if the user gave their consent before.

**Usage of Oauth: The** OAuth class has been discovered within the application during the analysis. This means that When the configuration of the OAuth service is compromised, attackers can exploit it to pilfer authorization codes or access tokens linked to other users' accounts. With a valid code or token in hand, the attacker gains potential access to the victim's data.

**Use of outdated vulnerable component: async@2.4.0: CVE 2021-43138**: the dependency async with version 2.4.0 has a security issue. The issue is identified by CVEs: CVE-2021-43138. CVE-2021-43138: A vulnerability in Async versions up to 3.2.1 for 3. x and up to 2.6.3 for 2. x (addressed in versions 3.2.2 and 2.6.4) allows a malicious user to gain privileges via the mapValues() method.

**Use of outdated vulnerable component: shelljs@0.8.4: CVE 2022-0144:** the dependency **shelljs** with version 0.8.4 has a security issue. The issue is identified by CVEs: CVE-2022-0144. The dependency **shelljs** Found in assets/index.android.bundle has a security issue (CVE-2022-0144) in which ShellJS is susceptible to Improper Privilege Management.

**Use of outdated vulnerable component: ws@6.1.4: CVE 2021-32640:** The dependency **ws** with version 6.1.4 has a security issue. The issue is identified by CVEs: CVE-2021-32640. The dependency **ws** Found in assets/index.android.bundle has a security issue which has A specially crafted value of the Sec-WebSocket-Protocol header can be utilized to considerably slow down a WebSocket server.

**ELF binaries do not enforce secure binary properties**: The application failed to enforce binary protections like RELRO, ASLR, No eXecute, and Stack canary, which are crucial memory protection techniques to guard against memory corruption exploitation.

**Unimplemented receiver Class Detected:** The analysis identified the receiver class declared in the manifest, such as <receiver android:name='ClassName'>, but no corresponding implementation for this class was found within the application's codebase. This discrepancy could lead to runtime errors or dysfunctional behavior during the application's execution.

List **of JNI Methods:** Improper use of the Java Native Interface (JNI) makes the application vulnerable to security flaws commonly found in other programming languages, such as memory corruption.

Table 2, depicts the type of vulnerabilities found in each of the digital bank applications. Generally, a total number of 32 vulnerabilities were identified in the bank applications using the Ostorlab tool for the analysis. Still, the specific number of vulnerabilities found in each bank app was indicated using a checkmate "✓" symbol to show the presence of a vulnerability in a particular bank application. In contrast, the **empty space** shows the absence of the vulnerability.

Table 2: The identified security vulnerabilities for the digital mobile bank applications

| Security Vulnerabilities | Carbon | Sparkle | Fundall | Mintyn | Eyowo | Fairmoney | Moniepoint | Vbank | Opay | Palmpay | ALAT | Kuda |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Insecure Object Serialization | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Undeclared permissions | ✓ | | | | | | | | | | | |
| Biometric Authentication bypass | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | |
| Attribute hasFragileUserData not set. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Attribute usesCleartextTraffic set | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ |
| Task Hijacking | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Services declared without permissions | ✓ | ✓ | | ✓ | | | | ✓ | ✓ | ✓ | ✓ | |
| Secret information stored in | ✓ | ✓ | | | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ |

 

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **the application** | | | | | | | | | | | | |
| **Use of WiFi API that contains or leaks sensitive PII** | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Unimplemented activity class detected** | ✓ | ✓ | | | ✓ | | | | | ✓ | | |
| **Application checks rooted device** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Debug mode disabled** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Backup mode disabled** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Exported activities, services, and broadcast receiver's list.** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Android Manifest** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Implementation of a WebViewClient** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Call to native methods.** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Broadcast receiver's dynamic registration** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Insecure network configuration setting** | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | |
| **Unimplemented service class detected** | | ✓ | | | | ✓ | ✓ | | ✓ | | | |
| **Implementation of a FileObserver** | | ✓ | | ✓ | | ✓ | ✓ | ✓ | | ✓ | | ✓ |
| **Insecure File Provider Paths Setting** | | | | | ✓ | | ✓ | ✓ | | | ✓ | |
| **Attribute requestLegacyExternalStorage Set** | | | | ✓ | ✓ | | | ✓ | | | | ✓ |
| **Deprecated API version** | | | | | ✓ | | | | | | | |
| **OAuth Account Takeover by hijacking custom schemes** | | | | | | ✓ | | | | | | |
| **Usage of Oauth** | | | | | | ✓ | | | | | | |
| **Use of outdated vulnerable component: async@2.4.0: CVE 2021-43138** | | | | | | | | ✓ | | | | |
| **Use of outdated vulnerable component: shelljs@0.8.4: CVE 2022-0144** | | | | | | | | ✓ | | | | |
| **Use of outdated vulnerable component: ws@6.1.4: CVE 2021-32640** | | | | | | | | ✓ | | | | |
| **ELF binaries do not enforce secure binary properties** | | | | | | | | | | | ✓ | |
| **Unimplemented receiver Class Detected.** | | | | | | | | | | | ✓ | |
| **List of JNI methods** | | | | | | | | | | | | ✓ |

### 4.2 The percentage of security vulnerabilities present in each digital bank mobile application

Viewing Table 3, the number of identified security vulnerabilities in each digital bank mobile application was listed as well as their level of percentage. Vbank mobile application is indicated to be a mobile app with the highest number of vulnerabilities with 68.75%. Sparkle and Eyowo bank mobile applications have the second highest number of security vulnerabilities with 62.5%. fairMoney and ALAT bank mobile applications have the third highest number of security vulnerabilities with 59.38%. Carbon also had a high number of security vulnerabilities with 56.25% while Opay, Palmpay, Kuda, and Mintyn mobile applications have 51.13% of security vulnerabilities. Moniepoint bank app has 50% of security vulnerabilities and lastly, Fundall had the least number of security vulnerabilities with 46.88%.

Table 3: The percentage of security vulnerabilities present in each digital bank mobile application

| Name of banks | Number of identified security vulnerabilities | Calculated Percentage |
|---|---|---|
| Carbon | 18 | 56.25% |
| Sparkle | 20 | 62.5% |
| Fundall | 15 | 46.88% |
| Mintyn | 17 | 51.13% |
| Eyowo | 20 | 62.5% |
| fairMoney | 19 | 59.38% |
| Moniepoint | 16 | 50% |
| Vbank | 22 | 68.75% |
| Opay | 17 | 51.13% |
| Palmpay | 17 | 51.13% |
| ALAT | 19 | 59.38% |
| Kuda | 17 | 51.13% |

### 4.3 The Security Vulnerability level identified in each digital mobile application

Table 4 shows the identified security vulnerabilities in each mobile app. For safe space, each bank is represented with alphabets. A represents Carbon Bank, B represents Sparkle Bank, C represents Fundall Bank, D represents Mintyn, E represents Eyowo Bank, F represents Fairmoney Bank, G represents Moniepoints Bank, H represents Vbank, I represents Opay, J represents Palmpay, K represents ALAT and L represents Kuda bank.

The vulnerabilities are classified into different levels of exposure and how risky they are. These include: High risk (red color), medium (deep orange), low (light yellow), potentially (ash color), Hardening (purple), important (light green), info (blue), secure (dark green), null (white)

**High-level:** this signifies that the level of risk at which the identified vulnerabilities can be exploited or exposed on the digital banks' applications is high**.** This is symbolized by the

 

**red color.** The vulnerabilities categorized as High level include Insecure File Provider Paths Settings found in Eyowo and Vbank applications. The vulnerability reveals that the application exposes a file provider with an insecure path setting at a high level which indicates security issues (it may harm the user or application infrastructure) and privacy issues (may result in intrusion into the user's private space, disclosing unauthorized information) in the applications. OAuth Account Takeover by hijacking custom schemes was another identified vulnerability found in the FairMoney bank application and categorized as High. This indicates that the vulnerability occurs when an application uses a custom scheme in the 'redirect_url' parameter during OAuth authentication, hence, showing security and privacy issues. Attackers can **bypass user interaction** by leveraging certain techniques like express authentication flow or using OAuth parameters that are meant to skip the consent prompt if the user gave their consent before. Use of outdated vulnerable components: async@2.4.0: CVE 2021-43138 and shelljs@0.8.4: CVE 2022-0144 found in the Vbamk application was another identified vulnerability that was categorized to be high. This means that dependency async with version 2.4.0 and dependency shelljs with version 0.8.4 has security issues that may result in harming the user of the application infrastructure.

**Medium-level:** this signifies that the level of risk at which the identified vulnerabilities can be exploited or exposed on the digital bank applications is medium (middle level)**.** This is symbolized by the **orange color**. The vulnerabilities categorized to be at medium level include insecure object serialization found across all the 12 digital bank applications. The vulnerability shows that all the applications used an insecure deserialization format that attackers can exploit to cause remote code execution which may result in modification of data as it brings up security issues (harming the user or application infrastructure). Undeclared permissions were another identified vulnerability found in the Carbon bank application and categorized to be at medium level. This means that the custom permission used in <activity> <service> <provider> <receiver> tags were not declared in <permission> tag and this may result in privacy violations or other security issues, such as an app accessing the user's location without their consent. Biometric Authentication bypass was also identified as a vulnerability found in Carbon, Sparkle, Fundall, Mintyn, Eyown, Vbank, Opay, Palmpay, and ALAT bank applications and categorized on a medium level. In this vulnerability, biometric authentication is done without CryptoObject or secretKey, making it vulnerable to biometric authentication bypass and resulting in security and privacy violations. Insecure File Provider Paths Setting was another identified medium-level vulnerability found in Moniepoint and ALAT bank applications. The vulnerability reveals that the application exposes a file provider with an insecure path setting at a high level which indicates security issues (it may harm the user or application infrastructure) and privacy issues (may result in intrusion into the user's private space, disclosing unauthorized information) in the applications. The Use of an outdated vulnerable component: ws@6.1.4: CVE 2021-32640 was identified as a medium-level vulnerability in the Vbank application. This means that the dependency `ws` with version `6.1.4` has a security issue, hence, harming the user of the application.

**Low-level:** this indicates that the level of risk at which the identified vulnerabilities can be exploited or exposed on the digital bank applications is low (low level)**.** This is symbolized by the **yellow color**. The low-level vulnerabilities include Attribute hasFragileUserData not set which was found all across the twelve selected bank applications. This vulnerability implies that the application does not explicitly set the attribute hasFragileUserData, that is, it does not allow developers to specify whether their app contains fragile user data that needs to be protected, raising privacy issues. Attribute usesCleartextTraffic set was also identified as a low-level vulnerability found in Carbon, Sparkle, Fundall, Mintyn, Eyown, FairMoney, Opay, Palmpay, Moniepoint, and Kuda bank applications. The vulnerability shows that The attribute specifies whether the app intends to use cleartext network traffic., hence, making it obvious for the attackers to violate the privacy and security space of the user since there's an absence of a secured network configuration. Task hijacking was another low-level vulnerability in all the selected digital bank applications. The vulnerability indicates that the application cannot protect against task hijacking, allowing malicious applications on the device to take over its foreground tasks. It may result in security and privacy issues, which can be targeted by malware and publicly available exploits. Attribute requestLegacyExternalStorage Set was identified as a low-level vulnerability found in Mintyn, Eyowo, Vbank, and Kuda bank applications. This indicates that the application sets the requestLegacyExternalStorage attribute, that is, It provides access to directories and various types of media files stored in external storage. Which attackers can take advantage of to harm the user and cause intrusion of privacy, disclosing unauthorized information? Lastly, the deprecated API version was identified as a low-level vulnerability found in the Eyowo bank app. The application sets the targetSdkVersion attribute which allows users of older API levels of Android, exposing users to security vulnerabilities.

**Potentially:** this implies that the level of risk at which the identified vulnerabilities can be exploited or exposed on the digital bank applications is on a potential level and the Ash color symbolizes this. This category does not indicate whether the vulnerability level is high, medium, or low but shows that they have the potential to occur as they need further investigation. The potentially grouped vulnerabilities include Services declared without permissions found in Carbon, Sparkle, Mintyn, Vbank, Opay, Palmpay, and ALAT bank apps. This shows that The declared services lack global permissions protection., potentially accessing sensitive information or performing privileged actions by attackers.

Secret information stored in the application was another potential identified vulnerability found in Carbon, Sparkle, Eyowo, FairMoney, Vbank, Opay, ALAT, and Kuda bank

 

apps. This shows that passwords, tokens, and other sensitive information are stored in the applications. This can potentially lead to the breach of confidentiality, Integrity, and availability of user's sensitive data. Use of WiFi API that contains or leaks sensitive PII was identified as a potential vulnerability found in Carbon, Sparkle, Fundall, Eyowo, FairMoney, Vbank, Opay, Palmpay, ALAT, and Kuda bank apps. This means that the applications used WI-FI API known to contain or leak sensitive PII (Personal Identifiable Information) used for tracking or monitoring. Backup mode enabled is also an identified potential vulnerability found in Sparkle, Fundall, FairMoney, Moniepoint, and ALAT bank applications. This means that the application is enabling backup mode, that is, allowing users to backup and restore data and settings from one device to Another, thereby, poses privacy threats, in which attackers can have unauthorized access to user data even on a different device if handled.

**Hardening:** this category does not specify whether the vulnerability level is high, low, or medium but indicates that the application needs to be further secured through implementing some security measures, as it poses security and privacy issues that can be exploited by the attacker. It is symbolized with **Purple color**. Vulnerabilities analyzed to be in the hardening category include unimplemented activity class detected which was found in Carbon and Sparkle bank apps. The vulnerability identified was the detected activity class in the manifest which did not correspond with the code in the application's codebase as this can lead to runtime error or malfunction of the application during execution causing security issues.

The insecure network configuration setting was also categorized to be in the Hardening level and was found in Sparkle, Fundall, Mintyn, Eyowo, FairMoney, Moniepoint, Vbank, and ALAT bank applications. The vulnerability indicates that the application either does not clearly state a network configuration setting or uses insecure settings., that is, there are no network security settings such as encryption of traffic, thereby, allowing attackers to have unauthorized access to user's sensitive data and intrude their private space. Another Hardening category was the unimplemented service class detected in Sparkle, FairMonney, Moniepoint, and Opay bank applications. The vulnerability identified was the detected service class in the manifest which did not correspond with the code in the application's codebase as this can lead to runtime error or dysfunctional behavior of the application during execution causing security issues. Also, ELF binaries do not enforce secure binary properties in the ALAT bank application. This means that the application does not implement binary protections (such as ASLR, NX, RELRO, and Stack canaries), that is, there's an absence of protection techniques to protect and mitigate the risk of memory corruption vulnerabilities like Buffer Overflow resulting to security issues. Lastly, "unimplemented receiver Class Detected" was found in the ALAT bank app which shows that the detected receiver class in the manifest did not tally with the code implementation in the application codebase and this discrepancy can lead to an error during application runtime.

**Secure:** this category is symbolized with a navy blue color. This category indicates some technical issues that may lead to security breaches even if the application seems to be secured. The identified issues include Application checks rooted devices which were found across all the selected bank apps. This shows that the application checks for the use of rooted or jail-broken devices. The absence of Jail-broken or Root detection is not a vulnerability, but its presence remediates the impact of Certain vulnerability classes or threats. "Debug mode disabled" was also identified across all the bank apps which indicates that The application is compiled with debug mode turned off. The debuggable attribute is not set and its default value is false, which can be manipulated by the attackers causing security issues. Finally, the "Backup mode disabled" attribute was found in Carbon, Mintyn, Eyowo, Vbank, Opay, Palmpay, and Kuda bank apps. The application is configured to disable backup mode, which prevents unauthorized access to sensitive data by preventing the full backup of applications, including private files saved on the /data partition. However, the allowBackup attribute is set to false.

**Important:** This is symbolized with **a green color**. It shows some technical issues that are to be noticed even if the applications seem safe to use. "Exported activities, services and broadcast receiver's list" was identified across all the selected bank apps as an important category that lists all the exported components in the application that are available to third-party applications, giving a means of access into the application for users. However, the exported activities, receivers, and broadcast definitions have technical issues that are flagged as important to view to prevent attackers' code tampering or manipulation.

**Info:** This is symbolized with a **sky-blue color** This category shows information about the application to the Android system, indicating some vulnerabilities such as "Android Manifest" in which the manifest file gives necessary information to the Android system before it can successfully run on the device. However, it has one issue in **AndroidManifest.xml,** and it is found across the bank apps. The "Implementation of a WebViewClient" was indicated across all the bank apps as Info. It contains the list of WebViewClient implementations of the application, though the entry is informative, there is a technical issue with the webview which may result in a security issue. The "Call to native methods" was identified across all the bank apps which involve the list of native methods calls, that is, a list of native Java methods was detected which could cause security issues. The "Broadcast receiver's dynamic registration "attribute is detected across all the bank applications. This indicates that one or more of the application's broadcast receivers are actively registered in the code and lack signature protection in the `AndroidManifest.xml` file., thus, giving attacker access and broadcast data that is not supposed to be exported to the public, leading to security issue for the user. Also, the "Implementation of a FileObserver" was identified in Sparkle, Mintyn, FairMoney, Moniepoint, Vbank, and Kuda bank applications. Another vulnerability was the "Usage of Oauth" found in FiarMoney, which means that when the

 

configuration of the OAuth service is compromised, attackers can exploit it to pilfer authorization codes or access tokens linked to other users' accounts. With a valid code or token in hand, the attacker gains potential access to the victim's data. The "List of JNI (Java Native Interface) methods" vulnerability was found in the Kuda bank application, which contains the list of JNI methods defined in ELF files and used by the application. This may lead to security issues if not properly used rendering the application susceptible to security flaws of the programming language.

**Null:** it is symbolized with No Colour (empty space). This means that no vulnerability is found at all.

Table 4: The level of security vulnerability in each digital bank mobile application

| Security Vulnerabilities | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Insecure Object Serialization | | | | | | | | | | | | |
| Undeclared permissions | | | | | | | | | | | | |
| Biometric Authentication bypass | | | | | | | | | | | | |
| Attribute hasFragileUserData t set | | | | | | | | | | | | |
| Attribute usesCleartextTraffic set | | | | | | | | | | | | |
| Task Hijacking | | | | | | | | | | | | |
| Services declared without permissions | | | | | | | | | | | | |
| Secret information stored in the application | | | | | | | | | | | | |
| Use of WiFi API that contains or leaks sensitive PII | | | | | | | | | | | | |
| Unimplemented activity class detected | | | | | | | | | | | | |
| Application checks rooted device | | | | | | | | | | | | |
| Debug mode disabled | | | | | | | | | | | | |
| Backup mode disabled | | | | | | | | | | | | |
| Exported activities, services, and broadcast receiver's list. | | | | | | | | | | | | |
| Android Manifest | | | | | | | | | | | | |
| Implementation of a WebViewClient | | | | | | | | | | | | |
| Call to native methods. | | | | | | | | | | | | |
| Broadcast receiver's dynamic registration | | | | | | | | | | | | |
| Insecure network configuration setting | | | | | | | | | | | | |
| Unimplemented service class detected | | | | | | | | | | | | |
| Implementation of a FileObserver | | | | | | | | | | | | |
| Insecure File Provider Paths Setting | | | | | | | | | | | | |
| Attribute requestLegacyExternalStorage Set | | | | | | | | | | | | |
| Deprecated API version | | | | | | | | | | | | |
| OAuth Account Takeover by hijacking custom schemes | | | | | | | | | | | | |
| Usage of Oauth | | | | | | | | | | | | |
| Use of outdated vulnerable component: async@2.4.0: CVE 2021-43138 | | | | | | | | | | | | |
| Use of outdated vulnerable component: shelljs@0.8.4: CVE 2022-0144 | | | | | | | | | | | | |
| Use of outdated vulnerable component: ws@6.1.4: CVE 2021-32640 | | | | | | | | | | | | |
| ELF binaries don't enforce secure binary properties | | | | | | | | | | | | |
| Unimplemented receiver Class Detected. | | | | | | | | | | | | |
| List of JNI methods | | | | | | | | | | | | |

### 4.4 The most common vulnerabilities discovered across all the digital bank's mobile applications

In a comparison of the OWASP top 10 vulnerabilities in mobile applications as explained in section 3.5 with the most common vulnerabilities identified across all the digital bank's mobile applications using the Ostorlab tool, it has been revealed that the vulnerabilities comply with the OWASP standards.

A total number of 16 vulnerabilities were discovered as common issues across all the digital bank's mobile applications. The vulnerability with the highest degree across all the banks is task hijacking which is a type of vulnerability that enables malicious users to take over the tasks of legitimate apps, tricking users into revealing sensitive information or performing unwanted actions. This means that across all the bank mobile applications, there's a presence and opportunity for malicious to hijack the legitimate sessions of the users to perform further attacks if the necessary solution is not applied. This is followed by the use of WiFi API that contains or leaks sensitive PII, which is the type of vulnerability that can be exploited by the attacker to perform further attacks, violating the legitimate user's privacy and security. Insecure object serialization is the third most prevalent vulnerability found across all mobile applications. This issue can lead to arbitrary remote code execution, modification of application logic, and altering of data, such as bypassing access control mechanisms. If timely

 

countermeasures are not taken. The fourth highest vulnerability identified across all bank mobile applications is the dynamic registration of broadcast receivers. These receivers are registered dynamically in the code and are not protected by signature permission in the `AndroidManifest.xml` file., thus, causing privacy issues for attackers to leverage on unless the appropriate security measures are taken. Biometric authentication is the fifth highest vulnerability which involves performing biometric authentication without CyptoObject or a secret key, making it vulnerable to biometric bypass.

The Unimplemented activity class detected is the next vulnerability in which no corresponding implementation for the activity class was found within the application's codebase and this discrepancy could lead to runtime errors or dysfunctional behavior during the application's execution. Another identified vulnerability is “Secret information stored in the application” in which leaking of secrets, passwords, and API keys can lead to a breach of the CIA (confidentiality, Integrity, and availability) of user’s sensitive data. The Exported activities, services, and broadcast receiver’s list is the next common vulnerability, which involves the exported components available to third-party applications, allowing entry points to the application, and can also be manipulated by the attackers due to identified technical issues. The next common vulnerability was the list of JNI methods, which involves improper use of the Java Native Interface (JNI) that exposes the application to security vulnerabilities common in other programming languages, such as memory corruption. The android manifest, application checks rooted device, Attribute hasFragileUserData t set, call to native methods, debug mode disabled, Implementation of a WebViewClient, attribute usescleartextTraffic set, insecure network configuration, backup mode disabled, implementation of a fileObserver and Insecure File Provider Paths Setting appear to be the common vulnerabilities at the low degree.

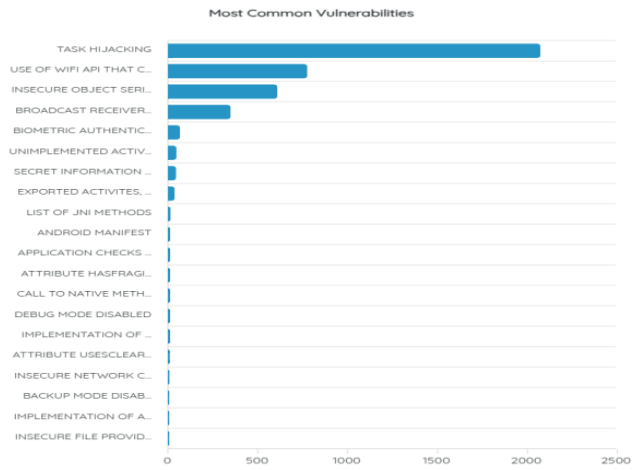


Figure 2: Most common vulnerabilities identified across all the Mobile applications

### 4.5 The overall Risk assessment (rating) of the bank's mobile applications

Table 5 shows the overall risk ratings of all the selected digital bank mobile applications after a thorough analysis has been conducted using the mobile application security testing tool (Ostorlab). The “target” column indicates the version of each mobile app downloaded from the Google Play Store. The “title” column refers to the name of each bank mobile app that was selected which was twelve in number, “the platform” column shows that all the apps were Android-based while the last column shows the overall risk level of each analyzed bank app. Eyowo, FairMoney, and Vbank mobile applications were identified to have a High overall risk level, which indicates that the presence of security vulnerabilities has a great impact and could easily be exploited if necessary security measures are not taken, hence, making their users susceptible to different forms of cyberattacks.

The other nine digital bank mobile applications: Carbon, Opay, Palmpay, Fundall, Sparkle, Kuda, ALAT, Moniepoint, and Mintyn banks were rated to be at Medium level, which means that the impact of the security vulnerabilities present in the mobile applications, can cause the users to experience minor injury or loss if it's not mitigated through necessary security measures.

Table 5: The overall risk rating of the digital bank mobile applications

| **Target (app version)** | **Title** | **platform** | **Overall Risk Level** |
|---|---|---|---|
| com.lenddo.mobile.paylater:9.3.6 (218) | Carbon | Android | Medium |
| ng. sparkle.Sparkle_android.prod:1.6.15 (160) | Sparkle | Android | Medium |
| com.fundall.io:3.8 (172) | Fundall | Android | Medium |
| com.mintfintech.app:1.4.9 (164) | Mintyn | Android | Medium |
| com.eyowo.android:3.1.20 (3120) | Eyowo | Android | High |
| ng.com.fairmoney.fairmoney:9.50.0 (445) | FairMoney | Android | High |
| com.moniepoint.personal:1.6.0 (26) | Moniepoint | Android | Medium |
| com.vfd.app:3.1.1-10 (101) | Vbank | Android | High |
| team.opay.pay:7.4.1.217 (7356432) | Opay | Android | Medium |
| com.transsnet.palmpay:5.5.0 (603292312 | Palmpay | Android | Medium |
| com.wemabank.alat.prod:4.3.7 (478) | ALAT | Android | Medium |
| com.kudabank.app:2.00278 (642) | Kuda | Android | Medium |

 

Table 6 illustrates the recommended countermeasures for each of the identified security loopholes in the digital bank's mobile applications. The countermeasures recommended for the identified vulnerabilities include the implementation of integrity checks including digital signatures on all serialized objects for the integrity of data (untampered data) and to avoid hostile object creation. In the case of undeclared permissions, all permissions should be well declared in the manifest file using <permission> element to avoid unnecessary actions that may be prone to illegal activities, also, there should be implementation of biometric authentication with CryptoObject usage to avoid unauthorized authentication. The hasFragieUserData attribute should be added to the androidmanifest file of the application and to ensure best practices for secure network communications within the mobile app, the usesCleartextTraffic attribute has been changed to "false". The issue of task hijacking can be avoided by ensuring that the application's activities adopt a randomly generated task affinity in which each activity runs in a different instance. Permission needs to be clearly stated as well as using the checkpermission method in the service methods, which can help in controlling access to services as requested by the users and also improving the security of the application. Confidential data of a user should never be included with the application itself, Rather, secure techniques for encoding, storing, and obtaining details for your services should be applied for accessing this information as requested, thus, minimizing the chances of malicious or illegitimate users having access to the user's information or using them to perform further cybercrime. There should be disabling of Unnecessary Wi-Fi data collection as well as using privacy-aware third-party libraries, which significantly improve the privacy of app users and also ensure compliance with best practices for data protection. The manifest file in the mobile application should be inspected to enhance the accuracy of declared classes as well as verify the existence of the indicated activity class within the application's code. The debuggable attribute should be properly set and its default value should be "True" while the "android:allowbackup" attribute should be set to enable or disable mode. The visual response to the mobile app users needs to be supplied about the loading status of the webpage in the WebView, improving the overall user experience. Also, export only those broadcast receivers that need to be opened by third-party applications while for others, a permission in the AndroidManifest.xml file with android:protectionLevel="signature" should be made to restrict usage to the application setting, hence, preventing access by third-party applications. It is paramount to confirm that the mobile app only communicates over encrypted channels, as well as configure a Network Security Configuration to disallow clear text traffic and enforce HTTPS for specified domains, hence, making the application more resistant to various security threats and ensuring a safer user experience. The manifest file needs to be inspected for accuracy in declared classes and verify the existence of the mentioned service class within the application's code as this helps in the smooth running of the activities within the application. There's a need to maintain a persistent connection to the FileObserver instance for continuous event notifications across different live objects, you should establish a shared reference accessible to all interested parties. Users need to be more careful about what files they share and only share files that are necessary and appropriate. In enhancing the protection of the app and user data on external storage, it's highly recommended to replace legacy storage mechanisms with scoped storage that provides better security and privacy of the app. The ""targetSdKVersion attribute should be set to a corresponding value of the latest Android API level, thus, minimizing the risk of vulnerabilities associated with older Android releases. A secure authentication mechanism such as OpenID connect (OIDC) can be adopted which helps in delivering the access token securely over HTTPS, storing the access token in a safe place using secure storage mechanisms, and configuring the access token with appropriate expiration times and scopes. Another recommendation involves using compilers such as GCC which enables some options that can help in protecting and detecting the memory space against some cyber-attacks such as buffer overflow which can be manipulated by illegitimate users finally native code in the app should be fortified against vulnerabilities such as memory corruption.

**Table 6: The countermeasures for the identified security vulnerabilities in the mobile applications**

| **Security Vulnerabilities** | **Countermeasures** |
|---|---|
| **Insecure Object Serialization** | Only allow Serialized Objects from Trusted Sources, limit serialization to basic types to avoid complexity and potential security risks, and implement Integrity Checks which include the use of digital signatures on all serialized objects to ensure their integrity and prevent hostile object creation or data tampering. |
| **Undeclared permissions** | There's a need to understand whether a declaration for permissions is needed in the app or using an alternative option to support the functionality in the app, hence, a permission should be well stated in the manifest file using <permission> element. |
| **Biometric Authentication bypass** | For native Android, Implement biometric authentication with CryptoObject usage. |
| **Attribute hasFragileUserData set** | The hasFragileUserData flag can be added to the application AndroidManifest.xml file. |
| **Attribute usesCleartextTraffic set** | By explicitly setting Android:usesCleartextTraffic to false and defining a comprehensive network security configuration, you ensure that your Android application adheres to the best tips for secure network communications. |
| **Task Hijacking** | Ensure that your application activities adopt a randomly obtained task affinity and that each activity runs separately, enhancing security and control over task management. |
| **Services declared without permissions** | Permission should be clearly stated as well as using the checkpermission method in the service methods, which can help in controlling access to services and also improving the security of the application. |

 

| | |
|---|---|
| **Secret Information stored in the application** | User's confidential data should never be included with the application itself. Rather, secure techniques for encoding, storing, and obtaining details for your services should be applied for accessing this information as requested. In preventing the risk of overbilling, consider implementing API key pinning or using authenticated APIs for services with potentially high usage costs. API key pinning helps to restrict the use of a key to a specific application by requiring a cryptographic signature, and it can be enabled by the service provider (e.g., Google Maps) |
| **Use of WiFi API that contains or leaks sensitive PII** | Unnecessary Wi-Fi data collection should be disabled as well as using privacy-aware third-party libraries, which significantly improve the privacy of app users and also ensure compliance with best practices for data protection. |
| **Unimplemented activity class detected** | Inspect the manifest file for accuracy in declared classes and verify the existence of the mentioned activity class within the application's code. Implement the necessary code for the declared activity or remove it if it is unnecessary. |
| **Application checks rooted device** | Root detection on Android can be conducted using the RootBeer library,which can use to a certain extent the **SafetyNet.** |
| **Debug mode disabled** | The debuggable attribute should be properly set and its default value should be "True" |
| **Backup mode disabled** | in the app manifest file, the attribute android:allowBackup should be set to enable or disable backup. |
| **Exported activities, services, and broadcast receiver's list** | This entry is informative |
| **Implementation of a WebViewClient** | There should be provision of visual response to the user about the loading status of the webpage in the WebView, enhancing the overall user experience. |
| **Call to native methods.** | The data provided to NewStringUTF must adhere to the Modified UTF-8 format. UTF-16 strings lack zero-termination. |
| **Broadcast receiver's dynamic registration** | Export only those broadcast receivers that need to be started by third-party applications while for others, a permission in the AndroidManifest.xml file with android:protectionLevel="signature" should be made to restrict usage to the application setting, hence, preventing access by third-party applications. |
| **Insecure network configuration setting** | Ensure that the mobile app only communicates over encrypted channels, as well as configure a Network Security Configuration to disallow clear text traffic and enforce HTTPS for specified domains. Furthermore, there should be implementation of certificate pinning using a custom Certificate Authority (CA). This involves verifying server certificates against your custom CA to prevent unauthorized connections, hence, making the application more resistant to various security threats and ensuring a safer user experience. |
| **Unimplemented service class detected** | Inspect the manifest file for accuracy in declared classes and verify the existence of the mentioned service class within the application's code. Implement the necessary code for the declared service or remove it if it is unnecessary. |
| **Implementation of a FileObserver** | There's a need to maintain a persistent connection to the FileObserver instance for continuous event notifications across different live objects, you should establish a shared reference accessible to all interested parties. |
| **Insecure File Provider Paths Setting** | Be cautious about what files you share and only share files that are necessary and appropriate. Don't share sensitive files or files that contain sensitive information. When using external-path, avoid using permissive settings like '.' as the path. Use the <grant-uri-permission> tag to control access to shared files. |
| **Attribute requestLegacyExternalStorage Set** | In enhancing the protection of the app and user data on external storage, it's highly recommended to replace legacy storage mechanisms with scoped storage that provides better security and privacy of the app. |
| **Deprecated API version** | Consider setting the android:targetSdkVersion attribute to a value corresponding to a recent Android API level. This can help ensure that the application benefits from security improvements and reduces the risk of vulnerabilities associated with older Android releases. |
| **OAuth Account Takeover by hijacking custom schemes** | To address these security loopholes, it is suggested not to use the custom scheme in redirecting authentication tokens. Options like App-to-app integration like Google Identity Services and Facebook Express Login for Android must be considered. |
| **Usage of Oauth** | In ensuring secure authentication using OpenID Connect (OIDC), best practices must be adhered to such as choosing the correct OAuth 2.0 grant type, delivering the access token securely over HTTPS, storing the access token in a secured place using secure storage mechanisms, and configuring the access token with appropriate expiration times and scopes. |
| **Use of outdated vulnerable component: async@2.4.0: CVE 2021-43138** | It is recommended for async to be updated to a version greater than or equal to 3.2.2. |
| **Use of outdated vulnerable component: shelljs@0.8.4: CVE 2022-0144** | It is recommended for shelljs to be updated to a version greater than or equal to 0.8.5 |
| **Use of outdated vulnerable component: ws@6.1.4: CVE 2021-32640** | It is recommended for **ws** to be updated to a version greater than or equal to 7.4.6. |
| **ELF binaries don't enforce secure binary properties** | Compilers such as GCC should be adopted which enables some options that can help in protecting and detecting the memory space against some cyber-attacks such as buffer overflow which can be manipulated by illegitimate users. |
| **Unimplemented receiver Class Detected.** | Inspect the manifest file for accuracy in declared classes and verify the existence of the mentioned receiver class within the application's code. Implement the necessary code for the declared receiver or remove it if it is unnecessary. |
| **List of JNI methods** | To fortify native code against vulnerabilities such as memory corruption, it's imperative to adhere to Secure Coding best practices. |

 

## 5. Conclusion and Future Scope

Despite various benefits obtained from adopting digital mobile applications in Nigeria by users in this digital era which helps to reduce stress in performing different financial

transactions anywhere and anytime, the comprehensive analysis conducted has revealed that there are a high number of vulnerabilities discovered in them. This has also been revealed by viewing the current OWASP Top 10 standard for Mobile Applications and comparing it with the most commonly identified vulnerabilities across all the twelve selected digital mobile applications using the Ostorlab tool. The digital mobile banks were selected based on the review of eight different researchers who talked about the most commonly used digital banks in Nigeria. The selected digital banks include Carbon, sparkle, kuda, fairmoney,moniepoint, ALAT, Vbank, palmpay, Opay, eyowo, fundall, and mintyn banks.

A total number of 32 vulnerabilities were discovered across all the selected digital banks while a total number of 16 vulnerabilities were the most common vulnerabilities and these include task hijacking, use of WIFI that contains or leaks sensitive PII, insecure object serialization, broadcast receiver, biometric authentication bypass, Unimplemented activity class detected, secret information stored in the application, exported activities services and broadcast receiver's list, list of JNI Methods, android manifest, application checks rooted device, Attribute hasFragileUserData set, Call to native methods, debug mode disabled, Implementation of a FileObserver, attribute usesCleartextTraffic set, insecure network configuration setting, backup mode disabled, Implementation of a WebViewClient and insecure File Provider Paths Setting.

The analysis reveals that the Vbank mobile application, also known as VFD, exhibits the highest vulnerability count, totaling 22, representing 68.75%., eyowo and sparkle mobile apps have a total number of 20 vulnerabilities with 62.5%, fairmoney, and ALAT banks have a total number of 19

Vulnerabilities each with 59.38%, carbon mobile app has a total number of 18 vulnerabilities with 56.25%, Mintyn, Opay, and Palmpay banks have a total number of 17 vulnerabilities with 51.13%, Moniepoint bank has a total number of 16 vulnerabilities with 50% while fundall total number of 15 vulnerabilities with 46.88%. The overall risk rating shows that Vbank (VFD). Eyowo and fairmoney banks were categorized at a high level while ALAT, kuda, sparkle, carbon, mintyn, opay, palmpay,fundall, and Moniepoint banks were categorized to be at a medium level. This clearly stated that that none of the mobile apps was found to be vulnerability-free and the one with the least vulnerabilities is fundall bank with 46.88%.

Having viewed through the analyzed results, therefore, it is very essential for users to be aware of the vulnerabilities in digital mobile applications and take necessary precautions such as having regularly updated versions of the digital bank mobile apps they are using to lessen the susceptibility rate of the app to the malicious users. User's confidential data such as the username and password should not be stored on the application as malicious attackers can make use of it when the particular Android mobile device gets stolen or lost to perform further cybercrime and can periodically change use strong passwords. Also, the network security configuration setting should contain an encryption method in which the traffic of data can be secured from malicious hackers and users should also reduce their level of total dependency on the digital bank mobile application for all forms of transactions. Therefore, further research works can be conducted in the area of adopting different mobile application security testing tools for deep analysis of more digital bank mobile apps to obtain better results and proffering more solutions and countermeasures that both users and application developers can adopt to minimize the tendency of malicious hackers hijacking the mobile or performing other cybercrimes with them through various means.

**Data Availability**
All details about the analysis report of each digital mobile application using Ostorlab are available on request.

**Conflict of Interest**
The author affirms that there is no undue influence with any party regarding the publication of this work.

**Funding Source**
None

**Authors' Contributions**
The author completed the entire work independently.

**Acknowledgments**
I sincerely thank God Almighty for His divine guidance, which illuminated my path and inspired my work. Additionally, I extend my gratitude to the International Journal of Scientific Research in Computer Science and Engineering for their invaluable feedback and constructive suggestions, which have remarkably increased the clarity and visibility of my research efforts.

## References

[1] M. Chauhan, G. Barapatr, A. Ghatge, R. Sabale, and P. S., "E – Authentication for Secure Net Banking," *International Journal of Scientific Research in Computer Science and Engineering*, Vol.**10**, Issue.**1**, pp.**15-18, 2022.**

[2] L. Garba, S. Ningi and A. Ahmed, "Impact of Website Design and Customer Technology Adoption on Customers Loyalty in the Nigerian Banking Industry," *World Academic Journal of Management*, Vol.**11**, Issue.**4**, pp.**42-47, 2023.**

[3] L. Wu, D. Yu, and Y. Lv, "Digital banking and deposit: Substitution effect of mobile applications on web services," *Finance Research Letters*, Vol.**56**, Issue.**104138**, pp.**1-5, 2023.**

[4] O. C. Ofodile, O. Odeyemi, C. C. Okoye, W. A. A. T. Oyewole, O. B. Adeoye and Y. J. Ololade, "DIGITAL BANKING REGULATIONS: A COMPARATIVE," *Finance & Accounting Research Journal*, Vol.**6**, Issue.**3**, pp.**346-371, 2024.**

[5] N. Koont, "The Digital Banking Revolution: Effects on Competition and Stability," *SSRN*, Vol.**1**, Issue.**1**, pp.**1-123, 2023.**

 

[6] S. P. Vilhena and R. D. Navas, "21THE IMPACT OF COVID-19 ON DIGITAL BANKING," *Journal of Entrepreneurial Researchers*, Vol.**1**, pp.**21-42, 2023.**
[7] S. Khan, R. Soni and K. J. Somaiya, "A study on Adoption of Digital Banking Services using Structured Equation Model," *Positif Journal*, Vol.**22**, Issue.**11**, pp.**126-148, 2022.**
[8] V. K. Sindhi, "Digital Banking in India: A Literature Review," *INTERNATIONAL JOURNAL FOR INNOVATIVE RESEARCH IN MULTIDISCIPLINARY FIELD*, Vol.**9**, Issue.**3**, pp.**103-106, 2023.**
[9] L. A. Bueno, T. F. A. C. Sigahi and R. Anholon, "Digital Banks in Brazil: Struggling to Reach the Breakeven Point or a New Evolution Wave?," *FinTech* , Vol.**2**, Issue.**3**, pp.**374-387, 2023.**
[10] E. M. Sasea and M. S. Sakmaf, "DIGITAL BANK LEGAL CHALLENGES: SECURITY PROTECTION AND LEAKAGE OF CUSTOMER PERSONAL DATA," *Awang Long Law Review*, Vol.**6**, Issue.**1**, pp.**245-250, 2023.**
[11] O. Alaba, O. Abass and E. Igwe, "Mobile Learning via Mobile Devices in Nigeria Higher Education: Usage Analysis Based on Utaut Model," *The Journal of the Southern Association for Information Systems*, Vol.**9**, Issue.**1**, pp.**64-80, 2022.**
[12] S. Sale, J. Godbole and V. Humbe, "New Emerging Trend in Payment System Special Reference to Electronic," *World Academic Journal of Management*, Vol.**11**, Issue.**1**, pp.**01-04, 2023.**
[13] M. Junker, M. Bo¨hm and H. Krcmar, "Advantages and disadvantages of mobile applications for workplace health promotion: A scoping review," *PLOS ONE*, Vol.**19**, Issue.**1**, pp.**1-22, 2024.**
[14] P. Falade and G. Ogundele, "Vulnerability Analysis of Digital Banks' Mobile Applications," *NDA Journal of Military Science and Disciplinary Studies*, Vol.**1**, Issue.**1**, pp.**44-55, 2022.**
[15] S. W. Asher, S. Jan, G. Tsaramirsis, F. Q. Khan, A. Khalil and M. Obaidullah, "Reverse Engineering of Mobile Banking Applications," *Computer Systems Science & Engineering*, Vol.**38**, Issue.**3**, pp.**266-278, 2021.**
[16] B. S. OKATAN and H. ÇAM, "Analysis of customer reviews for digital banking applications with text mining," *Gümüşhane University Journal of Science*, Vol.**14**, Issue.**1**, pp.**45-50, 2023.**
[17] B. S. Omotosho, "Analysing User Experience of Mobile Banking Applications in Nigeria: A Text Mining Approach," *CBN Journal of Applied Statistics*, Vol.**12**, Issue.**1**, pp.**77-108, 2021.**
[18] J. Zhu and M. Wang, "Analyzing the Effect of People Utilizing Mobile Technology to Make Banking Services More Accessible," *Frontiers in Public Health*, Vol.**10**, pp.**1-9, 2022.**
[19] S. Alhejji, A. Albesher, H. Wahsheh and A. Albarrak, "Evaluating and Comparing the Usability of Mobile Banking Applications in Saudi Arabia," *Information*, Vol.**13**, Issue.**12**, pp.**1-14, 2022.**
[20] E. K. Ghani, M. M. Ali, M. N. R. Musa and A. A. Omonov, "The Effect of Perceived Usefulness, Reliability, and COVID-19 Pandemic on Digital Banking Effectiveness: Analysis Using Technology Acceptance Model," *Sustainability*, Vol.**14**, Issue.**18**, pp.**1-16, 2022.**
[21] G. N. Wainaina, D. K. Kiyeng and N. Masese, "Enhancing Security Measures for Mobile Banking Applications: A Comprehensive Analysis of Threats, Vulnerabilities, and Countermeasures in Kenya Banking Industry," *International Journal of Computer Applications Technology and Research*, Vol.**12**, Issue.**8**, pp.**99-112, 2023.**
[22] M. A. Hassan, Z. Shukur and M. Mohd, "A Penetration Testing on Malaysia Popular e-Wallets and m-Banking Apps," *International Journal of Advanced Computer Science and Applications*, Vol.**13**, Issue.**5**, pp.**692-703, 2022.**
[23] D. O. Orucho, F. M. Awuor, C. Ratemo, and C. Oduor, "Security threats affecting user-data on transit in mobile banking applications: A review," *International Journal of Computer Engineering Research*, Vol.**9**, Issue.**1**, pp.**1-11, 2023.**
[24] S. A. Al-Delayel, "Security Analysis of Mobile Banking Application in Qatar," *arXiv preprint arXiv:2202*, pp.**1-7, 2022.**
[25] A. A. Ruth, O. F. Bukie and A. A. Ariyo, "Information Systems Research Methodologies: A Systematic Review on Cloud Adoption, Usage and Performance," *International Journal of Scientific Research in Computer Science and Engineering*, Vol.**11**, Issue.**5**, pp.**1-15, 2023.**
[26] S. Mazouzi, "Known Exploitable Vulnerabilities: Catching them all," 24 May 2024. [Online]. Accessed 24 May **2024.**
[27] T. Abiola, "10 Best Digital Banks in Nigeria: Usability, Features, Pros and Cons," 08 August 2023. [Online]. Accessed 08 August **2023.**
[28] Editorial, "Top digital banks in Nigeria (2024)," 09 April 2024. [Online]. Accessed 09 April **2023.**
[29] U. Anaga, "UNVEILING THE BEST ONLINE BANKS IN NIGERIA 2024," 01 January 2024. [Online]. Accessed 01 January **2024.**
[30] Business, "Reviving Savings Culture through Digital Banking," 09 April 2024. [Online]. Accessed 09 April **2024.**
[31] O. Willemin, "Best Digital Banks in Nigeria in 2022: Fees, Usability, Features, and More," 09 April 2024. [Online]. Accessed 09 April **2024.**
[32] Top5Editor, "Top 5 Best Virtual Banks in Nigeria (2024): Banking Made Seamless and Secure," 09 April 2024. [Online]. Accessed 09 April **2024.**
[33] S. Akintaro, "Here are 10 digital banks licensed by the CBN to operate as microfinance banks in Nigeria," 09 April 2024. [Online]. Accessed 09 April **2024.**
[34] J. Okwise, "Top 10 Best Online Banking Apps in Nigeria (Mobile App 2023)," 17 July 2023. [Online]. Accessed 09 April **2024.**
[35] D. F. Priambodo, G. S. Ajie, H. A. Rahman, A. C. F. Nugraha, A. Rachmawati and M. R. Avianti, "Mobile Health Application Security Assessment Based on OWASP Top 10 Mobile Vulnerabilities," in *2022 International Conference on Information Technology Systems and Innovation (ICITSI)*, Bandung, Indonesia, **2022.**

## AUTHORS PROFILE

**Grace Bunmi Akintola** She earned a B.Tech in Computer Science with a specialization in Cyber Security from the Federal University of Technology Minna, situated in Niger State, Nigeria, in 2016. Afterward, MSc in Computer Forensics and Cybersecurity was pursued from the University of Greenwich, London, United Kingdom, and graduated in 2021. These educational experiences have endowed her with a comprehensive understanding of cybersecurity principles, best practices, and forensic techniques. Currently, she serves as an Assistant Lecturer in the Department of Cyber Security at the Nigerian Defence Academy (NDA) in Kaduna, Nigeria. In this role, she has been actively involved in educating and impacting future cybersecurity professionals, fostering a cybersecurity awareness culture, and conducting field research. Her commitment to academic excellence is reflected in her continuous pursuit of knowledge and my dedication to her students. She is interested in research, scholarly writing, and other fields in cybersecurity, such as Network security, forensics, AI security, penetration testing, and others.

 