UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2026-019
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 09/02/2026 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Alioune DIALLO
Born on 20 January 1990 in Rufisque, (Senegal)

SECURITY OF ANDROID APPLICATIONS IN DEVELOPING REGIONS

Dissertation defence committee

Dr. Jacques Klein, dissertation supervisor
Professor, Université du Luxembourg

Dr. Tegawendé F. Bissyandé, Chairman
Professor, Université du Luxembourg

Dr. Samuel Ouya, Member
Professor, Université Numérique Cheikh Hamidou Kane, Sénégal

Dr. Gervais Mendy, Member
Professor, Université Cheikh Anta Diop de Dakar, Sénégal

Dr. Jordan Samhi, Member
Research Scientist, Université du Luxembourg

Abstract

Mobile apps are increasingly becoming the targets of attackers with the development of a huge number of mobile malware to exploit breaches worldwide. Governments and institutions are taking initiatives to block these practices, protecting critical institutions and safeguarding users. In developing countries, key sectors such as education, finance, agriculture, and healthcare increasingly rely on mobile applications running on handheld and low-cost devices to deliver essential services, enabling a leapfrogging effect in these sectors. However, these regions often face unique challenges, including limited cybersecurity infrastructure, lower digital literacy rates, and a higher prevalence of mobile-targeted cyber threats. Mobile application security has become a pressing concern where the impact of security breaches is amplified by the widespread use of low-end devices, limited supervision of pre-installed software, and the vulnerabilities found in critical applications. In Africa, for instance, mobile banking adoption is rapidly expanding, particularly within the West African Economic and Monetary Union (WAEMU) states, where financial institutions offer apps that enable users to transfer money, pay bills, and check balances at any time and from anywhere. Yet, this rapid proliferation of smartphones and applications raises critical security concerns. Poorly implemented security mechanisms during app development and deployment can expose users to significant privacy risks.

In this manuscript, we first conduct a systematic literature review to assess the current state of research on mobile application security within developing country contexts. Our investigation reveals a limited number of publications addressing this topic, suggesting a narrow academic focus. The findings underscore the need for more specialized research and tailored methodologies that address the unique security challenges of mobile ecosystems in developing regions.

Addressing the research gaps identified in the literature review, we examine pre-installed applications on low-cost Android smartphones widely distributed across Africa, including itel, Tecno, and Infinix devices. We developed *PiPLAnD*, a tool that extracts APK files directly from physical devices and performs static analysis on them. We analyze nine (9) low-cost devices to detect sensitive data leaks, manifest misconfigurations, and suspicious behaviors. The findings highlight that pre-installed software on low-cost smartphones can pose significant and widespread security and privacy risks.

Finally, we focus on financial applications from WAEMU financial institutions. Using static analysis, we evaluate 59 Android banking apps collected from 160 banks and financial institutions listed by the Central Bank of West African States. Our analysis reveals several security flaws introduced during development, some persist across multiple updates despite regular maintenance. To provide a broader

perspective, we compare these findings with banking apps from Europe, the United States, and other developing countries, revealing that WAEMU apps generally exhibit fewer critical issues but still present persistent weaknesses. Furthermore, we observe that WAEMU apps developed as local branches of foreign banks often inherit vulnerabilities from their parent applications while introducing new, context-specific issues.

In conclusion, this thesis provides a comprehensive view of mobile application security challenges in developing regions, with a particular focus on Africa. It highlights systemic issues arising from both pre-installed software ecosystems and financial applications, demonstrating the urgent need for stronger regulatory supervision, improved security practices during app development, and region-specific tools and methodologies to safeguard users' data and privacy. This work seeks to shed light on this invisible dimension of digital inequality and to contribute methods for assessing, measuring, and mitigating these risks. In doing so, it argues that true digital inclusion must go beyond access, it must ensure that the technologies enabling it are also trustworthy.

Acknowledgements

First and foremost, I would like to express my deepest and most sincere gratitude to my supervisor, Prof. Jacques Klein, for his invaluable guidance, constant support, and insightful feedback throughout this research journey. His expertise, patience, and high standards of academic excellence have profoundly shaped this work and my growth as a researcher. I am truly grateful for his trust, encouragement, and unwavering commitment.

I am also deeply thankful to my co-supervisor, Prof. Tegawendé F. Bissyandé, and my daily advisor, Dr. Jordan Samhi, for their continuous support and precious insights, which were vital in shaping and conducting my research. Their expertise and encouragement were invaluable throughout my Ph.D. journey.

I would also like to sincerely thank those who recruited me and gave me the opportunity to embark on this doctoral path, Prof. Samuel ouya and Prof. Gervais Mendy, as well as to all the members of the Labo LITA. Their confidence in my potential and their support were decisive in allowing me to pursue and complete this work. I am also grateful to Prof. Maïssa Mbaye and Prof. Cherif Diallo for their encouragement, guidance, and support throughout this process. Their trust and advocacy played a crucial role in creating the opportunity for my recruitment and in shaping the path that has led to this achievement.

My heartfelt appreciation goes to my co-authors and research collaborators, Dr. Jordan Samhi, Dr. Abdoul Kader Kaboré, Dr. Aleksandr Pilgun, Mrs. Aicha War, Mr. Moustapha Awwalou Diouf, Mrs. Anta Diop, and Dr. Steven Arzt. Our stimulating discussions, shared efforts, and collective reflections have greatly enriched this research. Working alongside such dedicated and talented colleagues has been both an honor and a continuous source of motivation.

I am deeply grateful to the members of the jury for accepting to evaluate this thesis and for dedicating their valuable time and expertise to its assessment. It is a privilege to have my work examined by such distinguished scholars.

I would also like to thank my colleagues of the TruX Research Group, espicially the the LuxWAyS team, for the intellectually stimulating environment, the constructive exchanges, and the camaraderie that made this journey both professionally rewarding and personally enriching.

Beyond the academic sphere, I wish to express my profound gratitude to the members of DETBN/UGB. Being part of this community has played a significant role in shaping both my character and my perseverance. I learned values of discipline, resilience, humility, and solidarity that have strengthened me throughout this demanding journey. The spiritual and moral support I received, sometimes quiet yet always sincere, has been instrumental in helping me remain focused and determined until the completion of this work. But, I reserve my special and heartfelt gratitude to

Seydina Ousmane Sylla, Mamadou Aliou Ba, Safiatou Diémé, and Marietou Diédhiou, as well as to all the members of this Dahira, for their invaluable support during the particularly difficult moments I experienced in their presence.

To my brothers from another mother, thank you for your encouragement, understanding, and constant presence. Your support brought balance, strength, and perspective during both the challenging and joyful moments of this journey.

My sincere appreciation also goes to my two extended families. To the family members who lovingly cared for my daughter and ensured her well-being, I am deeply grateful, especially to Cheikh Ibra Wade and Adji Coumba Wade. Your sister, my wife Aminata Wade (rest in peace), is the starting point of this success, and I dedicate it to her. Your support also granted me the peace of mind necessary to devote myself fully to my research. Without your dedication and generosity, balancing family life and academic responsibilities would have been far more difficult. To my other extended family, thank you for your constant presence, your wise advice, and your unwavering encouragement. Your guidance and support have been a source of reassurance and strength during moments of uncertainty.

My deepest gratitude goes to my family. To my father Amadou Baïlo Diallo and my mother Maïmouna Samba, thank you for your unconditional love, sacrifices, and steadfast belief in me. The values you instilled in me have been the foundation of all my achievements. I also wish to express my sincere appreciation to my uncle, Mamadou Saliou Diallo, as well as to my aunt Fatoumata Bintou Sonko, my brothers and sisters, for their support, guidance, and encouragement throughout my life.

Finally, and most importantly, to my beloved wife, Ndeye Seye Diagne: thank you for your patience, sacrifices, unconditional love, and constant support throughout this demanding journey. Your strength, understanding, and encouragement have been my anchor. This achievement is as much yours as it is mine.

ALHAMDOU LIL LAH!!!

Alioune DIALLO
University of Luxembourg
February 2026

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, we first give the context of this work by presenting the Android OS and its variants used in developing countries, particularly in Africa, and the problems. We then present the challenges of this research. Finally, we provide our research contributions followed by the roadmap of this thesis.

Contents

1.1 Contextualization

Technological advancement has radically changed our daily lives in recent decades, with the development of the Android operating system, providing tools such as tablets and smartphones. This operating system, which has become open-source, is the most widely used operating system worldwide, as illustrated in Figure 1.1. This has

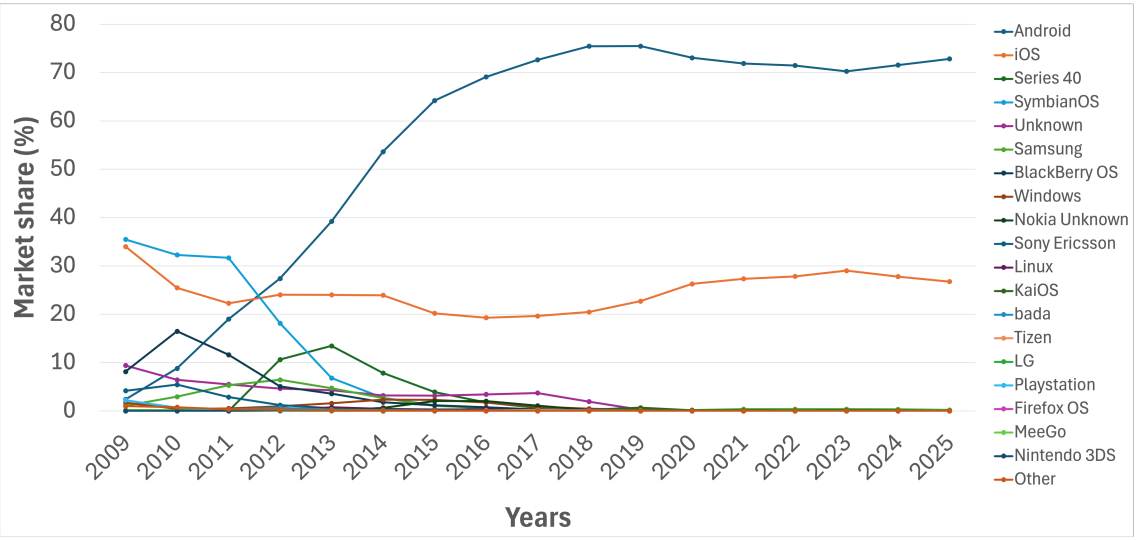


Figure 1.1: Trends in the mobile operating system market share from 2009 to 2025 [1].

led to rapid adoption of mobile use over the last decade [3, 4]. With 5.28 billion worldwide [4], smartphones have become ubiquitous and are used for a wide range ofen used for various tasks. In developing countries, mobile technology has enabled leapfrogging for mass adoption of critical digital services for accessing to education, health information, and mobile banking services that were previously out of reach for large segments of the population. Its adoption is forecasted to reach 87% by 2030 in Sub-Saharan Africa [5].

Yet, this rapid digital inclusion has unfolded in ecosystems dominated by low-cost Android devices. Indeed, the Android operating system has enabled affordable smartphones for millions of people worldwide. Devices typically ship with manufacturer- or vendor-installed systems and third-party apps, which can become a distribution vector for malware and privacy-invasive functionality, especially in developing regions [6]. Figure 1.2 shows that the Android OS is widely used in African countries, and Africa is leading the Android usage market share besides other continents. This rise is facilitated by the build of Android low-cost smartphones that remain widely used across the African continent [7] and have helped reduce the digital divide by expanding access to services [8, 9]. For example, roughly 20–22 million people in South Africa use smartphones, accounting for one third of the population [7]. Feature phones and inexpensive devices are still popular in Africa, which preserves the market opportunity for the expansion of low-cost Android devices. These low-cost devices, primarily used in developing countries, particularly in Africa, are built with Android Go Edition, a lightweight configuration of standard Android designed for entry-level devices with limited memory ($\leq 2\,\mathrm{GB}$) and storage [10]. With Android Go, low-cost Android initiatives have improved device accessibility [11]. It is not necessarily less secure than standard Android. These devices come with pre-installed applications

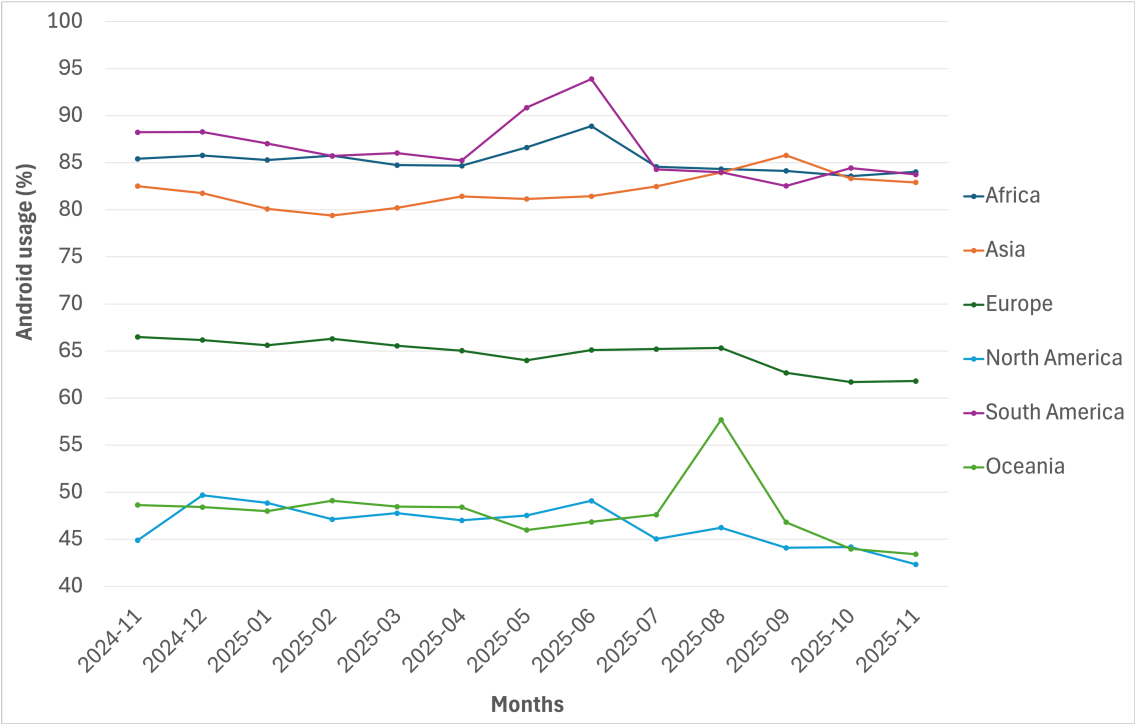


Figure 1.2: Trends in the Android OS market share by regions from November 2024 to November 2025 [1].

beyond users' control. In general, we are often unaware of the applications embedded in these devices or what they actually do. These apps, integrated at the system level, often operate with elevated privileges, lack transparency, and can access sensitive user data without consent or oversight The pre-installation process can be exploited to embed harmful apps that reach large user populations, as several of these apps are tailored to the specific needs of the region where the devices are distributed. Additionally, the digital financial services, including mobile money and mobile banking, emerge as powerful catalysts for the economic development of any country, particularly in African nations. The widespread adoption of mobile financial applications is a global phenomenon, with Sub-Saharan Africa leading the way in its innovative implementation [12]. This surge can be attributed mainly to the remarkable proliferation of low-cost smartphones, a trend projected to continue its upward trajectory. Recognizing the strategic advantage of widespread smartphone penetration, banks and financial institutions are leveraging this technology to extend banking services. This leap serves a dual purpose: broadening access to banking services for millions and empowering existing account holders with remote control over their finances, facilitating transactions and balance checks. In West African Economic and Monetary Union (WAEMU) countries, nearly all banks and financial institutions have adopted mobile applications, reaching millions of people who rely on these platforms for their daily financial transactions.

1.2 Challenges

Developing countries face unique challenges in many domains, particularly in the digital ecosystem [13], which directly impacts mobile app security. The limitations in connectivity, the low digital literacy, and reliance on outdated technologies contribute

to an elevated level of digital security risks. Despite that, there is not enough focus. The security of mobile applications in developing regions presents critical and underexplored challenges that demand focused investigation. While most existing research on mobile security focuses on mobile applications distributed through official markets in developed countries, users in developing regions face a very different reality, which highlights the need for a particular focus on mobile applications in these regions.

In African markets, low-cost Android devices dominate smartphone adoption, yet these devices frequently ship with pre-installed applications that operate with elevated privileges and receive minimal independent close examination. These technologies are used more and more, even though we have no real control over them. This situation creates a hidden attack surface where applications can compromise user privacy and system integrity. So, we want to determine the security status of the applications that these devices shipped. There is no existing work drawing a picture of the security of these applications. The main challenge is to have a dataset of these applications to know what kind of apps are embedded on low-cost devices. To do our work, we are forced to buy low-cost devices and extract apps from them to build a dataset. We build a pipeline that automates the extraction and the analysis porcess. Furtthermore, it was very challenging to get the complet list of methods used by apps to leak sensitive data since apps often rely on third-party library methods. We have extarcted all the methods from suspected apps and have used LLM to categorize them as Source or Sink to have a complete picture of the list of methods to use.

Concurrently, the growing adoption of Android devices facilitates the rapid proliferation of financial applications which are widely used in Africa. These applications often exhibit numerous and persistent security problems based on the challenges faced by African countries, exposing millions of users to financial fraud and data breaches [14, 15, 16, 17]. Almost all banks and financial institutions in WAEMU countries have deployed a mobile banking app to facilitate access to banking services. This needs a specific focus, allowing to improve app security and safeguarding users' money and financial institution infrastructure. As part of this work, the main challenge was to have the old versions of the apps allowing to analyze the security issues evolution. It was challenging to perform an manual analysis to confirm findings and avoid false positives, as well.

1.3 Research contributions

Considering these challenges and the security aspects mentioned above, the security of mobile applications is a critical concern in developing countries. Focusing on mobile application security in these contexts is vital for improving knowledge and enhancing defense against privacy violations, threats, and cyberattacks.

Through this dissertation, we make the following contributions:

- *(In)Security of Mobile Apps in Developing Countries: A Systematic Literature Review*. We systematically review the relevant literature concerning the security of mobile applications in developing countries. We group the studies into different types of studies to have a methodological landscape of the contributions proposed. We identify the different security issues explored, the analysis methods used, and the research directions taken to contribute to enhancing mobile app security. This study mainly highlights a significant research gap and very limited research directions in this domain in developing countries.

This work resulted in a research paper published in the Empirical Software Engineering (EMSE) journal in 2025.

- *On the security of pre-installed Android apps in low-cost devices.* We investigate different low-end Android devices mainly used in Africa. In this study, we propose a pipeline that extracts pre-installed APK files from physical devices and applies taint-based leak detection, pattern-driven behavior scanning, and manifest/component inspection to find data leaks, insecurely exported components, and suspicious behaviors. The dataset of pre-installed apps contains approximately 2050 APKs, and most of them are not available on Google Play. We identify several security concerns with these pre-installed applications, including misconfigurations in the AndroidManifest.xml files, suspicious app behaviors, and sensitive data leakage. We also identified some shipped applications that appear to exfiltrate sensitive data. These results indicate that pre-installed software on widely distributed, low-cost devices can pose real privacy and security risks to end users.
  This work resulted in a research paper presented at the 17th EAI International Conference on Africa Internet infrastructure and Services in 2025.
- *Security Assessment of Mobile Banking Apps in West African Economic and Monetary Union.* We provide an extensive analysis of mobile banking apps in WAEMU countries. This study identifies the most common security problems affecting WAEMU banking apps, assesses the security evolution in these apps, including whether concerns are effectively addressed in subsequent updates, offers a comparative analysis of the security posture of WAEMU banking apps in the global context, comparing them with the applications used by leading African and International banks, and highlights the security problems between the branch and parent apps. This contribution underscores the urgent need to enhance mobile banking app security to safeguard users and institutions in an increasingly digital financial landscape.
  This work resulted in a research paper presented at the 2025 Cybersecurity4D (CS4D) Conference and published in IEEE Xplore..

1.4 Roadmap

In Chapter 2, we present our systematic literature review (SLR) we conducted to identify relevant literature in the mobile application security domain in developing countries. Through this SLR, we offer an exhaustive survey of the field. This covers a broad range of pertinent research, thereby establishing a robust groundwork for future investigations. We provide a good and non-existent resource for researchers, practitioners, and policymakers aiming to enhance mobile app security in regions with unique challenges.

In Chapter 3, we analyze Android applications shipped on Android low-cost devices. This work only considers pre-installed applications from Android devices primarily used in Africa and provides an analysis of obvious security concerns. We consider the gap of existing works and build this study upon them to provide a more in-depth analysis.

In Chapter 4, we extensively investigate the security landscape of mobile banking applications in WAEMU countries. We identify prevalent security issues, establish a comprehensive threat model, analyze the evolution of security across different app versions, and compare the security posture of WAEMU banking apps with those from

the European Union (EU), the United States (US), and other developing countries (ODC). Moreover, we explore the security of subsidiary mobile banking apps and determine if they inherit parent app security issues.

Finally, we provide potential future works that are relevant to be explored in Chapter **??**, and we conclude this dissertation in Chapter 5.

2

(In)Security of Mobile Apps in Developing Countries: A Systematic Literature Review

In this chapter, we performed a systematic literature review to identify studies on mobile app security in developing countries. We have identified 25 primary studies and analyzed the existing research directions taken, the different security concerns addressed, and the techniques used by researchers to highlight or address app security issues. The main findings are: ❶ the literature includes only a few studies on mobile app security in the context of developing countries (only 25 publications found); ❷ among the different security concerns that researchers study, vulnerability detection appears to be the leading research topic; and ❸ FinTech apps are revealed as the main target in the relevant literature. Overall, this chapter highlights the significant room for developing further specialized techniques to address mobile app security in the context of developing countries.

This chapter is based on the work published in the following research paper [18]:

- Diallo, A., Samhi, J., Bissyandé, T.F., and Klein, J., (In)Security of mobile apps in developing countries: a systematic literature review. Empir Software Eng 30, 131 (2025). https://doi.org/10.1007/s10664-025-10689-z

Contents

2.1 Overview

Technological progress has enabled the introduction of powerful tools, including smartphones and tablets. These tools have witnessed remarkable global expansion during the last decade [3, 4]. Smartphones have become ubiquitous and are now used for a wide variety of tasks and services [19].

A developing country, also known as an underdeveloped, low-income, or middle-income country, is generally defined as a country that has not yet achieved full maturity in terms of economic and industrial development [20]. The United Nations Development Programme (UNDP) utilizes the Human Development Index (HDI)[1] score to assess whether a country falls into the category of developing or developed. In developing countries, mobile technology has played a pivotal role in accelerating access to essential digital services.Recent forecasts estimate that smartphone adoption in Sub-Saharan Africa will reach 87% by 2030[2].

While the rapid proliferation of smartphones offers numerous opportunities and benefits [21], it also raises significant security and privacy concerns due to the growing range of associated risks [22]. Mobile security threats and app vulnerabilities can have severe implications, especially for critical digital services.

Mobile applications frequently handle highly sensitive data — including user credentials, financial details, medical information, and location data [23, 24], making them prime targets for cyberattacks [24]. At the same time, malicious actors increasingly develop mobile malware specifically aimed at users in developing regions. In fact, the majority of the ten countries most affected by mobile malware in 2022 were developing nations [25], where over 10% of mobile devices are estimated to be infected [26].

Developing countries face distinct challenges, including limited connectivity, the prevalence of low-cost but insecure devices, and low levels of digital literacy. These challenges directly affect the security of their mobile ecosystems [13] and contribute to an elevated level of digital insecurity. Developers may unintentionally introduce vulnerabilities during app development, while outdated technologies and weak update mechanisms increase existing risks. For instance, mobile financial applications in developing countries face recurring security challenges related to encryption, authentication, and the management of sensitive assets such as keys and credentials [15]. Furthermore, a variety of fraudulent and technical attacks have been observed in mobile financial systems, including data theft through malware, account hijacking via SIM swap attacks, unauthorized MSISDN modifications, Denial of Service (DoS) attacks, man-in-the-middle interceptions targeting mobile money systems, and Short Message Service (SMS) Spoofing [16, 17].

Given these realities, mobile application security represents a critical concern in developing countries. Focusing on mobile app security in these contexts is essential not only to protect users' privacy and personal data but also to enhance resilience against emerging cyber threats and large-scale attacks.

Several secondary studies have been performed in the context of developing countries. Hoque et al. [27] conducted a review of studies related to mobile health applications, evaluating the quality of evidence reporting in the mobile health literature using the Mobile Health Evidence Reporting and Assessment (mERA)

[1] `https://hdr.undp.org/data-center/human-development-index#/indicies/HDI`

[2] Source: `http://www.osiris.sn/En-Afrique-subsaharienne-le-taux-d.html`

checklist, as recommended by the World Health Organization (WHO). In this work, the authors reveal a low level of familiarity with the mERA checklist among researchers in developing countries, and most mHealth studies do not adequately meet the essential criteria for evidence reporting. Msweli et al. [28] reviewed the existing literature about the adoption of mobile banking services among the elderly in developing countries, focusing on the enablers and barriers. The paper highlights that while mobile banking has become prevalent in both developed and developing nations, research focusing on its adoption among the elderly is limited. The main barriers identified include security concerns, trust and privacy issues, a lack of personalization, and limited technical knowledge among the elderly. On the other hand, significant enablers include the perceived ease of use of mobile banking applications, perceived value, convenience, and consumer attitudes. Malik [29] reviewed empirical research on Internet and mobile banking adoption in developing countries. The review focuses on the factors influencing the adoption of these technologies and the methodologies used in the studies. It also discusses the variables that have been extended in the Unified Theory of Acceptance and Use of Technology (UTAUT) model. In the study, the author highlights the directions, the most used analysis tools, and the main indicators of behavioral intention used in the publications.

Other studies have investigated the security of mobile apps. For instance, Martinez-Perez et al. [30] evaluate the current state of privacy and security in mobile health (mHealth) apps, focusing on reviewing existing laws regulating privacy and security in the European Union (EU) and the United States (USA), analyzing the corresponding academic literature, and proposing recommendations for app designers to ensure compliance with current security and privacy legislation. The review identified several research lines, including proposals for secure systems, authentication techniques, and privacy aspects in Body Sensor Networks (BSNs). Key findings in this study highlighted the need for enhanced security mechanisms in mHealth apps, as well as the importance of user consent and data protection. Other authors reviewed existing literature to identify and analyze the security issues associated with the use of mobile educational apps [31]. The goal is to highlight the security challenges and propose solutions to address these issues, thereby enhancing the security and usability of mobile educational apps. Their review identified several key security aspects in the use of mobile educational apps, including reliability, integrity, trust, privacy/confidentiality, and availability. They suggested several ways to address these security challenges. Despite the efforts made to address the security of mobile apps in developing countries [32, 33, 34, 35], we have identified five secondary studies that investigate the security of mobile apps; however, none of them focus on this specific context. This suggests that a comprehensive study is needed to provide an overview of the current state of knowledge in this domain.

To address this gap, we conducted a systematic literature review (SLR). Our primary goal is to examine existing works on mobile app security, with a specific focus on mobile apps in developing countries. This chapter provides actionable and impactful information for those addressing mobile application security issues in regions where resources are scarcer and challenges are more pronounced. In this work, we have excluded countries with high and very high HDI and focused solely on those with medium and low HDI, based on the classification provided by the UNDP in its Human Development Report[3]. This focus ensures that our findings are

[3]`https://hdr.undp.org/sites/default/files/2021-22_HDR/HDR21-22_Statistical_An`

relevant to contexts where the need for robust mobile app security measures is most critical. Our work aims to identify current research directions and highlight gaps in the literature, thereby pinpointing areas that require further investigation and study. Through a systematic review of existing studies, we provide an exhaustive survey of the field. This covers a broad range of pertinent research, thereby establishing a robust groundwork for future investigations. By doing so, we aim to contribute to the development of more secure mobile applications in developing countries.

The main contributions of this work are as follows:

- We conduct a systematic review of the literature on the security of mobile apps in developing countries. By following a systematic process, we identify relevant studies, extract data, and synthesize the findings.
- We report on the type of research conducted. We categorize the studies based on their research focus (e.g., user study, app analysis, study of the development framework, app security testing, or study that focuses on proposing a design & implementation solution). This analysis clarifies the methodological landscape and provides guidance for future research directions.
- We report on the security concerns addressed. We report the specific vulnerabilities and threats explored in the literature. By highlighting the security issues most relevant to developing countries, our review helps prioritize research efforts and tailor solutions to local challenges.
- We report on the type of analysis performed. We classify the methods used to assess mobile app security. This provides a valuable reference for researchers and practitioners seeking effective evaluation approaches.
- We identify research directions that must be followed to contribute to the development of more secure mobile apps in developing countries. By suggesting specific areas for improvement, we guide future research efforts. This encourages the development of specialized techniques that directly address the needs of these regions.

By combining these elements, we provide a valuable resource for researchers, practitioners, and policymakers seeking to enhance mobile app security in regions with unique challenges.

Data Availability. The datasets used in the current study are available in our online repository [36].

2.2 Background and Related Works

2.2.1 Background

Mobile app security pertains to the measures implemented to safeguard mobile apps from various forms of threats, including hacking, mobile malware, data breaches, privacy violations, and other malicious activities[4]. In practical terms, mobile app security includes various aspects integrated during the app design process, such as:

- Ensuring that only authorized users can access the app by verifying their identity (e.g., using passwords, biometrics, or two-factor authentication).
- Controlling users' actions within the app based on their roles and permissions.

nex_HDI_Table.xlsx

[4]https://fraudwatch.com/blog/what-is-mobile-app-security-including-8-application-security-tips/

- Ensuring APIs used by the app are secure (e.g., validating input, using tokens).
- Protecting against API abuse and unauthorized access.
- Avoiding hardcoding secrets in the source code.
- Securing data transmitted between the app and servers using secure protocols such as HTTPS.
- Encrypting sensitive data when storing on the device.
- Storing sensitive data (such as passwords, tokens, or keys) securely within the app
- Implementing runtime security controls (e.g., obfuscation, anti-tampering mechanisms) to prevent reverse engineering and code modification.
- Keeping the app up-to-date with security patches and bug fixes.

Ignoring these aspects can lead to various security risks, including data breaches, privacy violations, and mobile malware attacks. Specifically, a data breach in the context of mobile apps refers to a security incident where unauthorized parties gain access to sensitive or confidential information stored within the app, including personally identifiable information, credit card numbers, and medical records [37]. Privacy violations in mobile apps refer to instances where an app mishandles sensitive user data, potentially compromising user privacy. Besides, mobile malware is malicious software that can exploit vulnerabilities to compromise the user's data security and privacy on the device or other installed apps. Mobile apps can contain exploitable risks and unsecured entry points that threat actors, including malware, can leverage.

Given that mobile apps often handle sensitive information, such as financial records, personal health data, and location data. Numerous security concerns, including insecure communication, insecure data storage, improper usage of cryptography, and improper platform usage [38], can compromise the security of this information, leading to identity theft, financial losses, or reputational damage.

Adversaries can exploit these security issues using various methods, including man-in-the-middle attacks, authentication attacks, device theft, SMS interception, and more [17].

2.2.2 Related Works

To our knowledge, no systematic literature review focuses on mobile app security in developing countries. However, several reviews, investigations, and surveys have been performed on other topics in the context of developing countries.

Hsu et al. [39] gave an overview of Chinese mobile health (mHealth) apps in December 2015 by investigating the most downloaded apps from Android and iOS. In their study, authors aimed to understand the current state of the Chinese mHealth market, focusing on medical-related apps categorized by ten different medical initiatives rather than general health apps. For each app, they analyzed the main service offered, mHealth initiative, disease and specialty focus, app cost, target user, Web app availability, and emphasis on information security. The findings revealed that the primary mHealth initiatives targeted by the apps reflect Chinese patients' demand for access to medical care. Disease-specific apps are also representative of disease prevalence in China, with the most common disease-specific apps focusing primarily on diabetes, hypertension, and hepatitis management. Most apps were found to be free and available on both iOS and Android platforms. The paper also underlines the lack of information about users' data security since developers did not mention the purposes for which users' data could be used.

Latif et al. [40] presented a study that underlines the factors hindering the use of mobile health in developing countries by reviewing various mobile health initiatives, their impacts, and their healthcare challenges. The review highlights several challenges that hinder the successful deployment of mHealth, including infrastructural limitations, the need for strategic partnerships, and cultural and language issues. The authors emphasized the importance of leveraging emerging technologies such as the Internet of Things (IoT) and artificial intelligence (AI) to enhance mHealth adoption. Additionally, the paper features a case study on Pakistan to validate the findings and illustrate the practical implications of mHealth in a specific context. This case study demonstrates that tailored approaches, considering local cultural contexts, are essential for effective mHealth implementation.

Abdullaev et al. [41] performed a state-of-the-art survey of mobile banking. They aimed to classify and analyze mobile banking services' challenges and security issues in Uzbekistan. They explored the security issues related to the Wireless Application Protocol (WAP). With this protocol, customers can use bank services through the Internet. However, data is not well encrypted at one stage of the communication using WAP. They also identified other risks and issues related to the authentication process, SMS banking, and virus attacks, in which attackers can use many techniques to steal users' sensitive information. They presented many security risks that can compromise the mobile banking system. In their study, authors also found that 60% of customers do not use mobile banking technology.

Azeez and Lakulu [42] performed a literature review on mobile government (m-government) in developing countries focusing on identifying the benefits, challenges, and critical success factors for the successful implementation of m-government from both government and citizen perspectives. The authors analyzed various studies on the success of m-government and categorized the success factors based on their impact on the successful implementation of m-government. The findings reveal increased efficiency of governmental activities, cost reduction in organizations, and accessibility of government services as benefits of m-government, as well as the offer of real-time information and improvement of citizen participation in governmental activities. However, the study identified several m-government challenges, including security and privacy, technical limitations, infrastructure limitations, and interoperability. The paper concluded that while m-government offers significant opportunities for improving government services in developing countries, it also presents several challenges that need to be addressed to ensure successful implementation.

Hoque et al. [27] present a review of studies related to mobile health applications between 2013 and 2018. Through this review, they aimed to evaluate the quality of evidence reporting by using mobile health Evidence Reporting and Assessment (mERA), a checklist developed by the World Health Organization (WHO). This review highlights the application level of evidence reporting as recommended by WHO. Authors found that researchers and mobile health intervention designers from developing countries have limited familiarity with the mobile health Evidence Reporting and Assessment checklist. They also noticed that the majority of studies fail to meet the essential evidence-reporting criteria outlined in the checklist. Furthermore, design science-based methods and theory-based frameworks are rarely applied in the development of mobile health interventions. Finally, they found that most mobile health interventions are not prepared for interoperability or integration into existing health information systems. Overall, the study reveals that most studies do not

properly apply WHO's recommendations.

There is also another state-of-the-art in which Rahman et al. [43] explore the current situation of mobile banking in Bangladesh. The research objectives are to identify the problems and challenges that customers face in mobile banking services and to observe the future prospects of mobile banking in the country. The paper identified several problems related to security (financial loss across virus attacks, fraudulent activities, privacy leakage, etc.), time (because of late payments or any other reason, there is time lost), network (poor network), performance (server break down), financial (lose of money because of mistakes made during money transaction), and lack of banking knowledge. Authors also argued that issues may arise because of compatibilities (difficulty to operate with another device when the first breaks down, unable to access service in some places because of poor internet connection). According to them, trusting mobile banking is an important factor that may give issues such as fairness, capability, and beneficence. Despite these challenges, the paper highlighted the prospects of mobile banking, including benefits for phone operator companies, increasing job scopes, no service charge, increased purchasing power, and easy money transfer. The authors concluded that if these issues can be solved in the future, mobile banking will lead to a greater impact on the banking economy of Bangladesh.

Msweli et al. [28] provided a study that explores the enablers and barriers to mobile banking and mobile commerce among the elderly, particularly in developing countries, by reviewing the existing literature focusing on literature from 2009 to 2019. The findings reveal that there are significant gaps in research concerning the elderly and mobile banking, particularly in developing countries. Key barriers identified include security concerns, lack of trust, and limited technical knowledge among older adults, which hinder their adoption of mobile banking services. Conversely, enablers such as the perceived ease of use and the potential for improved quality of life through mobile commerce were noted. The study concludes that there is a pressing need for further research to address these gaps and to develop tailored mobile banking solutions specifically to the needs of the elderly, thereby enhancing their financial inclusion in the digital era. Similarly, Pankomera and van Greunen [44] systematically reviewed the opportunities, barriers, and adoption factors of mobile commerce (m-commerce) services for the informal sector in Africa. This paper seeks to provide comprehensive insights into how m-commerce can benefit the informal sector, the challenges faced, and the factors influencing its adoption. The study identified several barriers, including the limited network coverage and broadband infrastructure, high costs of mobile devices and services, resistance due to illiteracy, lack of trust and traditional business practices, and finally, a lack of legal and regulatory frameworks to support m-commerce. The adoption factors identified from the literature include network coverage and availability of electricity, knowledge of m-commerce and its benefits, perceived usefulness and ease of use, affordability of m-commerce solutions, confidence in the security of m-commerce transactions, accessibility of financial services, the impact of social and cultural factors on adoption, and government policies and regulations promoting m-commerce. Furthermore, the authors identified several benefits and opportunities of m-commerce, such as the creation of new services, increased revenue, reduced operational costs, and enhanced productivity and market access, particularly in the agriculture and fishing sectors.

Malik [29] performs a review article in which he studies publications that talk

about Internet and mobile banking adoption from 2015 to 2020 using the Unified Theory of Acceptance and Use of Technology (UTAUT) model in developing countries. This study highlights the directions, the most used analysis tools, and the main indicators of behavioral intention used in the publications. It also reveals that most publications focused on factors affecting Internet banking adoption (54%). This study argues that the extended UTAUT model is mostly used in publications. Indeed, as presented, this model is used by 67% of the publications among the 54% internet banking and by 70% among the 46% mobile banking publications. This is similar to the study that reviews the literature regarding mobile health adoption in developing countries [45]. In this study, Aljohani et al. identify the methodologies used in existing research, the significant factors influencing adoption, and the gaps in the literature, particularly regarding the use of qualitative and mixed methods. The authors conducted a systematic review of the literature by evaluating various studies published between 2010 and 2020. The findings revealed a limited number of studies on m-health adoption in developing countries, with a notable concentration of research in China and Bangladesh. The Technology Acceptance Model (TAM) was frequently used, focusing primarily on technological and individual factors, while other health-related factors and theories were underexplored. Furthermore, most studies employed quantitative methodologies, with only one study utilizing a qualitative approach and none using mixed methods. The authors emphasized the need for more qualitative and mixed-method studies to provide richer insights into the adoption of m-health applications and to better understand the cultural and health impacts of these technologies.

All of these mentioned studies are completely different from ours because we have mainly focused on investigating studies that have been conducted in this area.

2.3 Methodology

In this work, we have followed the systematic literature review guideline proposed by Keele [46].

Figure 2.1 illustrates our research process. It can be broken down as follows:

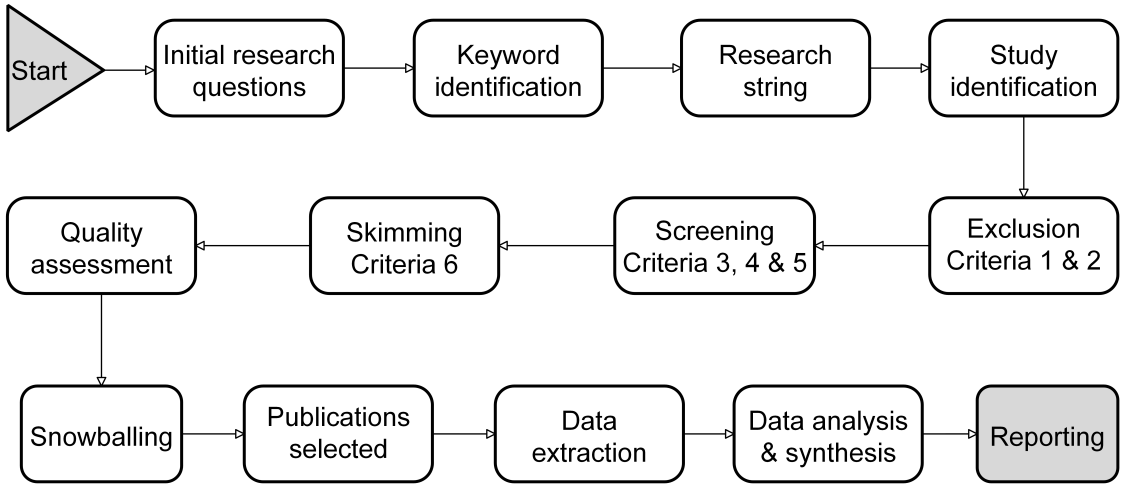


Figure 2.1: Overview of the methodology followed in this SLR

- We initiated the process by formulating precise research questions, serving as a foundation for identifying relevant publications.

- Guided by our research questions, we identified pertinent keywords.
- Previous keywords were instrumental in pinpointing a substantial body of relevant studies, culminating in constructing a comprehensive search string.
- Leveraging the search string developed in the prior step, we conducted systematic searches in four reputable digital libraries: IEEE Xplore, ACM Digital Library, Springer, and ScienceDirect.
- Having gathered publications from the digital libraries, we applied predefined exclusion criteria to filter out the most relevant publications.
- To further refine our selection, we meticulously reviewed the titles and abstracts of each paper, adhering to specific criteria. Only studies meeting these criteria were retained.
- We established a list of criteria to assess the quality of publications by removing poor-quality publications.
- We also employed backward and forward snowballing of selected publications to identify any potentially missed relevant ones.
- We extracted data from selected publications, allowing us to answer research questions.
- Then, we based on the data extracted to perform data analysis to guide the interpretation of the results.
- The final step reports our findings to the research community.

2.3.1 Research questions

To provide an understanding of the publications produced by researchers to address the specific challenges of mobile app security, we have formulated a set of research questions (RQs). These RQs serve as the backbone of our investigation, and each sheds light on key aspects of this research field.

RQ0: In which venues are these studies published? Understanding where these studies are published provides a better understanding of the spread of knowledge about mobile app security in developing countries. This question overviews the conferences and journals where researchers have published their work.

RQ1: What are the research directions around mobile app security in developing countries? Mobile app security research can take various directions. Investigating these directions in developing countries points out the areas where researchers have chosen to concentrate their efforts. This knowledge informs us about the prevailing concerns and priorities in this context.

RQ2: What are the security concerns addressed in the literature? The security of mobile apps is a fundamental concern worldwide. As the use of mobile apps continues to grow in developing countries, understanding the specific security issues that researchers address becomes crucial. This question unveils the most frequent security concerns relevant to mobile apps in these regions.

RQ3: Which apps are covered in the literature? This research question enables us to identify which of the multitude of application categories available are covered in the literature for developing countries. Recognizing the categories publications have focused on helps identify the gaps and areas needing further exploration.

RQ4: What techniques are used to detect security issues? Researchers employ various techniques to detect security issues in mobile apps. By unveiling these techniques, we gain insights into the strategies employed to identify security

issues in the context of developing countries.

RQ5: What are the specific characteristics of malware and the techniques used to compromise devices? This research question aims to identify the various types of Android malware targeting developing countries. This will not only categorize the types of malware, but will also examine the techniques and procedures used, the active threat actors in these regions, the malware payloads, the most targeted information assets, and the attacks vectors.

RQ6: What motivated researchers to investigate mobile app security in developing countries? This question elucidates the various motivations mentioned in the literature, providing an understanding of the factors that have inspired researchers to investigate mobile app security in developing countries.

Each research question plays a key role in clarifying the multifaceted landscape of mobile app security in developing countries, contributing to a more informed and comprehensive understanding of this critical field.

2.3.2 Search strategy

Getting relevant publications began by crafting a search string and executing our search across four well-regarded repositories.

Search string. Our search string was constructed based on a list of keywords

Table 2.1: Keywords used in this SLR.

Category	Search keywords
Device	android; mobile; smartphone; "smart phone"; iphone; ios; "portable device"; app*
Security	security; malware; vulnerabilit*; weak*; exploit; flaw; breach; leak*; malicious; phishing; ransomware; trojan; attack; compliance; crypto*; forensic; "reverse engineering"; encryption; threat; hack*; "privacy violation"
Target	"developing countr*"; "developing world"; "low-income countr*"; "middle-income countr*"; "low and middle-income countr*"; Africa*; "south Asia*"; "least-developed countr*"; **Country names**[5]

derived from our previous research questions. These keywords were categorized into three groups, as displayed in Table 2.1: "Device," "Security," and "Target." Each line in the table represents a category with multiple corresponding keywords. For each category, we created a string, denoted as c_i (with i representing the category number), using a disjunction of its keywords, such as $c_1 = k_1$ OR k_2 OR k_3 OR ... OR k_n.

[5]Egypt; Libya; Angola; Benin; Botswana; "Burkina Faso"; Burundi; Cameroon; "Cabo Verde"; "Central African Republic"; Chad; Comoros; Congo; "Democratic Republic of the Congo"; "Ivory coast"; Djibouti; "Equatorial Guinea"; Eritrea; Eswatini; Ethiopia; Gabon; Gambia; Ghana; Guinea; "Guinea Bissau"; Kenya; Lesotho; Liberia; Madagascar; Malawi; Mali; Mauritania; Mauritius; Mozambique; Namibia; Niger; Nigeria; Rwanda; "St. Helena"; "Sao Tome & Principe"; Senegal; "Sierra Leone"; Somalia; "South Sudan"; Sudan; Tanzania; Togo; Uganda; Zambia; Zimbabwe; Albania; Armenia; Azerbaijan; Belarus; "Bosnia & Herzegovina"; Georgia; Kosovo; Macedonia; Moldova; Montenegro; Serbia; Ukraine; Belize; "Costa Rica"; Cuba; Dominica; "Dominican Republic"; "El Salvador"; Grenada; Guatemala; Haiti; Honduras; Jamaica; Montserrat; Nicaragua; Panama; "St. Lucia"; "St. Vincent and the Grenadines"; Bolivia; Ecuador; Guyana; Paraguay; Peru; Suriname; Venezuela; Afghanistan; Bhutan; Cambodia; Kyrgyzstan; "Lao People's Democratic Republic"; Maldives; Mongolia; Myanmar; Nepal; Pakistan; Tajikistan; "Timor Leste"; Turkmenistan; Uzbekistan; Jordan; Lebanon; "Syrian Arab Republic"; "West Bank and Gaza Strip"; Yemen; "Cook Islands"; Fiji; Kiribati; "Marshall Islands"; Micronesia; Nauru; Niue; Palau; "Papua New Guinea"; Samoa; "Solomon Islands" ; Tokelau; Tonga; Tuvalu; Vanuatu; "Wallis & Futuna"; Philippines; Morocco; Bangladesh; India

We obtained three strings (c_1, c_2, c_3). We combined these three strings in our search string as a conjunction, resulting in the final search string, i.e., final_string = c_1 AND c_2 AND c_3. This final string was used in online repositories for our systematic search.

Well-known repositories. We conducted our SLR across four repositories: IEEE Xplore, ACM Digital Library, ScienceDirect, and Springer. The choice of these repositories is motivated by the fact that they are the most important and widely used in the scientific community. We employed Advanced Search for the first three repositories to refine our results, focusing on each publication title and abstract. In Springer, we cannot focus our search on the title or the abstract; we performed a classic search across all metadata and filtered the results to computer science publications. To account for specific constraints related to search fields, we subdivided the main search string into multiple strings before executing the search. Notably, IEEE Xplore allows a maximum of 25 keywords in one clause. ScienceDirect restricts string searches to a maximum of 8 boolean operators. While ACM Digital Library and Springer did not explicitly describe these constraints in their documentation, we opted to apply the same limitations as IEEE Xplore to ensure optimal results.

Handling large results. Situations occur where our searches returned more than 1,000 items, exceeding Springer's display limit. We employed a Python script to scrape all findings. Additionally, Python script-based verification was used. As mentioned above, in IEEE Xplore and ACM DL, we specifically focused our search on publication titles and abstracts to refine the results. However, in Springer, we couldn't narrow down to titles and abstracts directly. Therefore, we initially conducted a general search, yielding numerous publications. Subsequently, we refined our search by targeting the titles and abstracts of the retrieved publications, ensuring consistency in the filtering process. In our Python script-based verification, we constructed a search string similar to the one used in IEEE Xplore. For titles, the search string looked like this: **(('android' in title.lower() or 'mobile' in title.lower() or ...) and ('security' in title.lower() or 'malware' in title.lower() or ...) and ('Country_1' in title or 'Country_2' in title or ...))**. For abstracts, the search string was as follows: **(('android' in abstract.lower() or 'mobile' in abstract.lower() or ...) and ('security' in abstract.lower() or 'malware' in abstract.lower() or ...) and ('Country_1' in abstract or 'Country_2' in abstract or ...))**. Unlike IEEE Xplore and ACM DL, ScienceDirect's search may include publications that do not contain the complete search string in the title or abstract, This also requires the Python script-based verification for additional checking to filter results effectively.

By employing these methods, we ensured comprehensive coverage and accuracy in our search across these repositories.

2.3.3 Exclusion criteria

Following our search across cited repositories, we encountered a multitude of publications, including many irrelevant to our SLR due to the use of generic keywords. To curate a consistent dataset of relevant publications, we applied a set of exclusion criteria:

1. **Remove duplicate publications.** We employed a Python script to identify and remove duplicated entries based on title, abstract, author names, and year of publication.We found around 58 212 of duplicated publications that we

removed.

2. **Exclude books, thesis reports, and non-English written papers**. Our search on ACM Digital Library returned several books and thesis reports that were not aligned with our SLR objectives, as well as publications that were not written in English. Therefore, we removed these entries (around 414), focusing on journal articles and conference papers.

3. **Filter out unrelated publications**. Some keywords in our search string yielded papers unrelated to mobile apps. We manually reviewed the remaining publications' titles and abstracts, excluding those deemed irrelevant. This step further refined our dataset, discarding approximately 98% of the results.

4. **Exclude non-security-focused publications.** Our dataset contains publications exclusively discussing mobile apps without addressing security. During the manual review of publications' titles and abstracts, we removed those publications.

5. **Remove non-developing-country papers.** During the manual review, we excluded publications that did not explicitly mention "developing country or related terms (e.g., "low-income countries," "emerging economies", name of a developing country, etc.). After this step, we have 88 remaining publications.

6. **Comprehensively review each paper.** We ensured each publication explicitly focuses on mobile application security in developing countries by reviewing each paper entirely. The authors have discussed any publication and decided to exclude 62 irrelevant publications when a consensus was reached.

By implementing these exclusion criteria, we systematically filtered and refined our dataset, ensuring that the publications included in our SLR were directly related to mobile app security in developing countries. At this stage, our dataset contains 26 publications. Figure 2.2 presents an overview and offers a clearer picture of the publication selection process.

2.3.4 Quality assessment

Given that there is no universally accepted definition for 'quality' in the context of a study, the assessment of a study's quality can vary significantly depending on the purpose of the study [47, 48]. The quality assessment process serves as an additional layer of scrutiny after applying general exclusion criteria. This step is crucial in mitigating biases that may arise from studies of low quality. In our study, we perform the quality assessment to provide a better quality of publications. This enables us to address the research questions more effectively and guides the analysis and interpretation of the results. Based on [47, 48], we have defined 5 quality assessment (QA) criteria that help us ensure optimal publication quality. These criteria are detailed in Table 2.2. We restrict our consideration to those publications that fulfill at least 3 of these QA criteria. As a result of this selection process, we have excluded 3 from the 26 remaining publications. These 3 papers indeed only fulfill 2 of the criteria listed in Table 2.2.

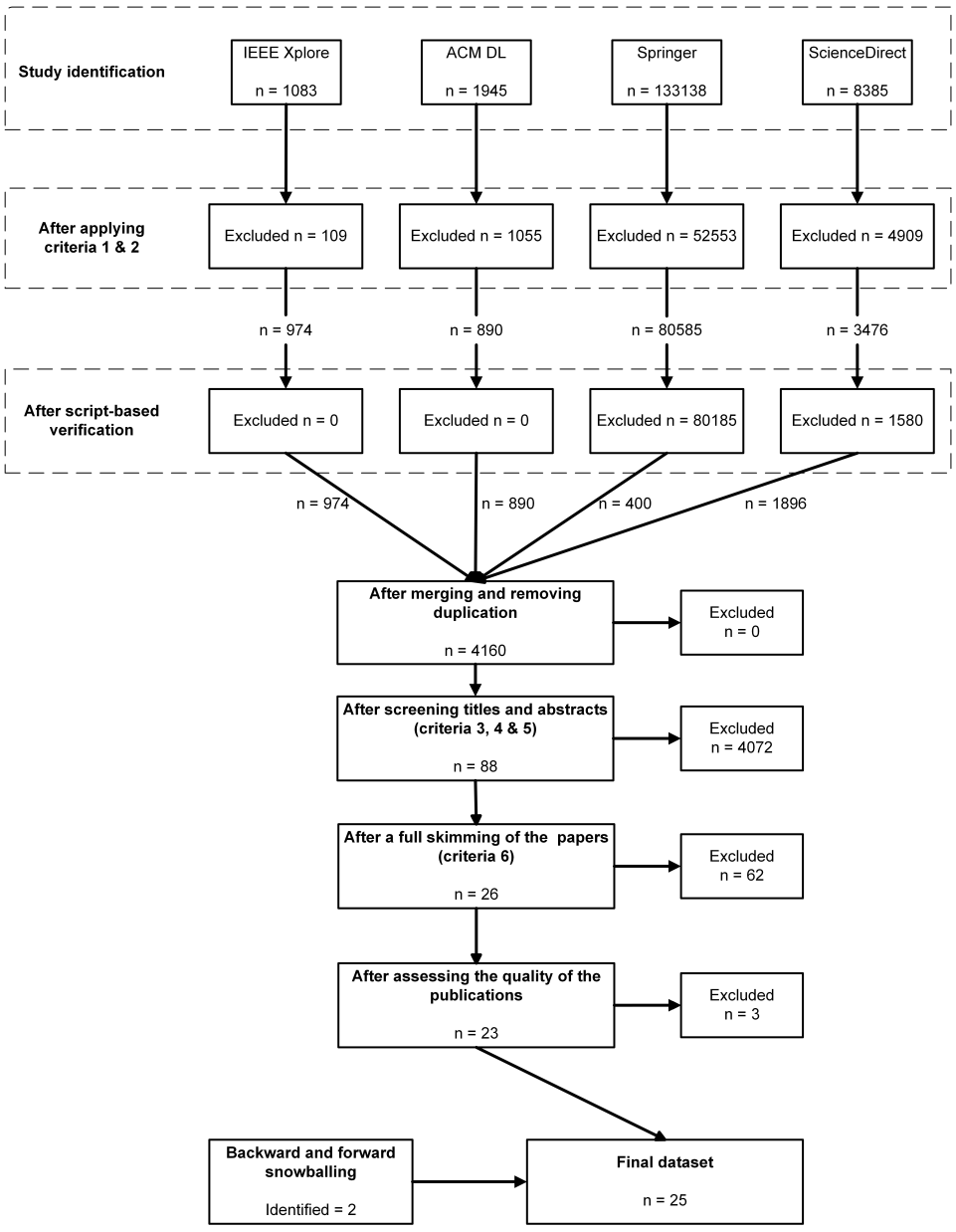


Figure 2.2: Process of publication selection

Table 2.2: Criteria used for assessing the quality of the publications.

	Assessment criteria
QA1	Is the paper clearly motivated?
QA2	Are the objectives of the paper clearly stated?
QA3	Does the paper provide a detailed description of the procedures used?
QA4	Are the results of the paper clearly presented and interpreted in the context of the objectives?
QA5	Does the paper discuss the key contributions?

2.3.5 Snowballing

Since we have focused on the most important and widely used repositories in the scientific community, we could probably miss relevant publications published in other repositories. To deal with that, we initiated a backward and a forward snowballing process once we identified the primary publications relevant to our SLR. This involved a combination of automated scripts and manual efforts to collect references from these publications (backward snowballing) and publications having cited our dataset publications (forward snowballing). Using these collected publications, we utilized scripts to verify and remove any publications that did not align with the focus of our SLR.

Following this initial screening, we manually reviewed the remaining publications. If we encountered relevant publications that were not part of our originally selected primary studies, we incorporated them into our dataset. This iterative process ensured the comprehensive inclusion of pertinent literature in our SLR. We retrieved two additional publications from the backward snowballing.

Table 2.3: The full list of the primary publications selected.

Year	Venue type	Venue	Publication title
2023	Conference	ICOEI	Effective Security Testing of Mobile Applications for Building Trust in the Digital World [49]
2023	Conference	ACM SIGCAS/SIGCHI	Evaluating Mobile Banking Application Security Posture Using the OWASP's MASVS Framework [50]
2022	Conference	IEEE S&P	"Desperate Times Call for Desperate Measures": User Concerns with Mobile Loan Apps in Kenya [51]
2022	Journal	HPT	Adoption of Covid-19 contact tracing app by extending UTAUT theory: Perceived disease threat as moderator [52]
2022	Conference	ICKES	User's Perception on Security and Privacy in Using Crypto Currency Trading Application in India [53]
2022	Conference	ICCCNT	State of Survey: Advancement of Knowledge Environmental Sustainability in Practicing Administrative Apps [54]
2021	Journal	HPT	Understanding digital contact tracing app continuance: Insights from India [55]
2021	Journal	RCS	Determining factors and impacts of the intention to adopt mobile banking app in Cameroon: Case of SARA by afriland First Bank [56]
2021	Conference	ICSCCC	Security Issues of Unified Payments Interface and Challenges: Case Study [57]
2021	Conference	CT-RSA	Mesh Messaging in Large-Scale Protests: Breaking Bridgefy [58]
2020	Conference	AFRICOMM	Analysis of the Impact of Permissions on the Vulnerability of Mobile Applications [32]
2020	Conference	InterSol	Vulnerability Analysis in Mobile Banking and Payment Applications on Android in African Countries [34]
2020	Conference	COMPASS	We Don't Give a Second Thought Before Providing Our Information: Understanding Users' Perceptions of Information Collection by Apps in Urban Bangladesh [59]
2020	Conference	COMS2	Signature Based Malicious Behavior Detection in Android [60]
2020	Conference	USENIX Security	Security Analysis of Unified Payments Interface and Payment Apps in India [61]
2019	Conference	ICSIoT	A Comparative Study of User Data Security and Privacy in Native and Cross Platform Android Mobile Banking Applications [62]
2019	Conference	ICCSA	Forensic Analysis of Mobile Banking Apps [35]
2019	Journal	MAT	Forensic analysis of mobile banking applications in Nigeria [63]
2018	Conference	RTEICT	Integrating OAuth and Aadhaar with e-Health care System [64]
2017	Conference	ICTD	A Study of Static Analysis Tools to Detect Vulnerabilities of Branchless Banking Applications in Developing Countries [33]
2017	Journal	TOPS	Mo(Bile) Money, Mo(Bile) Problems: Analysis of Branchless Banking Applications [65]
2017	Conference	ICEEG	Side-Effects of Permissions Requested by Mobile Banking on Android Platform: A Case Study of Morocco [66]
2017	Conference	NSysS	Vulnerability detection in recent Android apps: An empirical study [67]
2016	Conference	ANTS	On the MitM vulnerability in mobile banking applications for android devices [68]
2016	Conference	ACM DEV	Let's Talk Money: Evaluating the Security Challenges of Mobile Money in the Developing World [17]

Table 2.3 presents the full list of the selected paper.

2.3.6 Data extraction

Table 2.4: The list of the data extracted in the selected publications.

Extracted information	Description	Corresponding RQ
Venue name	The name of the venue where the paper has been published.	
Venue Type	The type of the venue where the paper has been published.	RQ0
Venue location	The location where the venue held.	
Author institution	The origin of the publication (the affiliations of the authors).	
Type of contribution	The type of contribution researchers performed.	RQ1
Secuirty issues	The issues of security researchers addressed in their publications.	RQ2
Category of app	The categories (according to GooglePlay) of app researchers studied in their works.	RQ3
Issue detection technique	The techniques and methods used to detect issues on the apps.	RQ4
Malware characteristics	The types of malware identified, their procedure and compromise techniques, their payloads, threat actors and targeted information assests	RQ5

In this section, we present the information extracted to address the research questions mentioned earlier. Table 2.4 provides an overview of the data extracted from the selected publications. To answer RQ0, we collected metadata for each publication from the repository websites where they were published. These repositories offer comprehensive information about the publications. The extracted data include details related to the venues and the authors' affiliations. Next, we thoroughly read each publication to extract relevant information. Specifically:

- For RQ1, we identified the types of contributions conducted by researchers.
- To address RQ2, we examined the security concerns they tackled.
- For RQ3, we categorized the apps they focused on.
- To answer RQ4, we analyzed the various techniques used to detect and address security issues in these apps.
- Finally, to answer RQ5, we analyzed publications and extracted information related to the malware types, their payloads, the techniques and procedures used to compromise the phones, the active threat actors, and the information assets that are attacked the most in this region.

We have decided to group the selected publications into types of contributions after reading and getting results from the extraction of data. More details have been provided about the extracted data in the next section.

2.3.7 Data analysis and synthesis

In the previous section, we extracted several data to address research questions. Based on these data, our analysis involves systematically addressing each research question with the appropriate analysis technique.

2.3.7.1 Quantitative analysis

In this section, we particularly focus on quantifying the data extracted from the previous section. We use descriptive statistics to summarize the metadata. For example, we count the number of publications per venue type, geographical distribution of venues, and frequency of publications from different institutions. We then create frequency distributions and visualizations to see the prevalence of each type of contribution, each detection technique, and the distribution of app categories. Finally, we quantify the occurrence of different security aspects and malware types, techniques, and threat actors.

2.3.7.2 Open coding (Qualitative analysis)

To apply open coding, we read through the papers, identified key concepts (e.g., security), and assigned "labels" (e.g., permissions, or cryptography) to these concepts. Then, we grouped similar "labels" into broader categories (e.g., security concerns). To ensure that they accurately represent the data, we reviewed and refined the categories. By applying this process to the papers, we identified several key categories, including app categories, geographic focus, contribution types, analysis methods, security concerns, and analysis focus. This process helps in organizing and interpreting the data, making it easier to draw meaningful insights and address the RQs.

2.4 Results

Table 2.5: Mobile app origin area (**N.S**: Not Specified).

Paper	App country origin	Contient
[17]	N.S	Dev. countries
[32]	Burkina Faso	Africa
[33]	N.S	Dev. countries
[34]	N.S	Africa
[35]	Nigeria	Africa
[49]	India	Asia
[50]	N.S	Africa
[51]	Kenya	Africa
[52]	India	Asia
[53]	India	Asia
[54]	Bangladesh	Asia
[55]	India	Asia
[56]	Cameroon	Africa
[57]	India	Asia
[58]	Belarus, Zimbabwe	Africa, Europe
[59]	Bangladesh	Asia
[60]	India	Asia
[61]	India	Asia
[62]	Ghana	Africa
[63]	Nigeria	Africa
[64]	India	Asia
[65]	N.S	Dev. countries
[66]	Morocco	Africa
[67]	Bangladesh	Asia
[68]	India	Asia

In this section, we present and interpret the study results and answer the research questions. The study consists of identifying research works concerning the security of mobile apps used in the context of developing countries. We have identified 25 publications investigating mobile app security.

All have, indeed, investigated mobile apps used in developing countries. In Table 2.5, we map publications with the context of the study to show where the apps

(a) Publications frequencies from different areas.

(b) Publications per venue type.

Figure 2.3: Overview of publication distribution.

addressed are used.

2.4.1 RQ0: In which venues are these studies published?

This section details the distribution of publications according to venue type, geographic origin, and conference locations. In this work, a paper is considered to originate from a specific place if all authors' affiliations match that location. As illustrated in Figure 2.3a, 76% of publications come from a developing country.

We have found that 80% (20 out of 25) of the publications are presented at conferences, with the remaining 20% in journals, as illustrated in Figure 2.3b. Almost every venue accounts for one publication, except one journal venue for two.

Given the preponderance of conference publications in the literature, it is noteworthy that researchers often present their work at conferences held in the country of their institutions or in a developing country. Indeed, approximately 75% of conference publications (corresponding to 15) originate from developing country institutions. Among these publications, three [35, 50, 66] were not presented at conferences held in developing countries, as illustrated in Table 2.6. Conversely, non-developing countries contribute a few conference publications [17, 51, 58, 59, 61], with one taking place in a developing country [17].

Among the journal publications, one is identified as non-developing countries' contribution [65].

Answer to RQ0

Most studies have been presented at conferences held in developing countries. Nevertheless, mobile app security in developing countries has attracted the interest of developers located outside these countries.

2.4.2 RQ1: What are the research directions around mobile app security in developing countries?

This section outlines the types of contributions performed by researchers examining mobile app security in developing countries. As presented in Table 2.7, the literature primarily focuses on five main types: user studies, development framework

Table 2.6: Overview of publications and venues.

Year	Paper	Venue	Location	VHIDC*	1st author's institution	ILIDC**	Co-authors' institutions (# of authors)
Conference							
2019	[62]	ICSIoT	Accra, Ghana	Yes	Kwame Nkrumah Univ. of Sci. and Tech., Kumasi, Ghana	Yes	Kwame Nkrumah Univ. of Sci. and Tech., Kumasi, Ghana (1)
2017	[33]	ICTD	Lahore, Pakistan	Yes	Information Technology University, Pakistan	Yes	Information Technology University, Pakistan (2) University of Washington, USA (1)
2019	[35]	ICCSA Russia	Saint Petersburg,	No	Federal University of Technology, Minna, Nigeria	Yes	Federal University of Technology, Minna, Nigeria (3) Covenant University, Ota, Nigeria (1)
2016	[17]	ACM DEV	Nairobi, Kenya	Yes	University of Washington, USA	No	University of Washington, USA (3) Cornell University, New York, USA (1)
2020	[59]	COMPASS	Ecuador	No	Utah State University, USA	No	University of Dhaka, Bangladesh (1) Jadavpur University, India (1) Utah State University, USA (1) University of Toronto, Canada (2)
2020	[34]	InterSol	Nairobi, Kenya	Yes	Université Joseph Ki-Zerbo, Ouagadougou, Burkina Faso	Yes	Université Joseph Ki-Zerbo, Ouagadougou, Burkina Faso (3)
2022	[51]	IEEE S&P	San Francisco, CA, USA	No	Geoge Washington University, Washington, DC, USA	No	Geoge Washington University, Washington, DC, USA (2)
2019	[32]	AFRICOMM	Porto-Novo, Benin	Yes	Université Joseph Ki-Zerbo, Ouagadougou, Burkina Faso	Yes	Université Joseph Ki-Zerbo, Ouagadougou, Burkina Faso (4)
2017	[66]	ICEEG	Turku, Finland	No	University Cadi Ayyad, Marrakesh, Morocco	Yes	University Cadi Ayyad, Marrakesh, Morocco (2)
2022	[53]	ICKES	Chickballapur, India	Yes	Symbiosis University, Indore, India	Yes	Symbiosis University, Indore, India (2)
2016	[68]	ANTS	Bangalore, India	Yes	University of Hyderabad, Hyderabad, India	Yes	Centre for Mobile Banking, IDRBT, Hyderabad, India (2)
2017	[67]	NSysS	Dhaka, Bangladesh	Yes	Bangladesh University of Engineering and Technology	Yes	Bangladesh University of Engineering and Technology (2)
2022	[54]	ICCCNT	Kharagpur, India	Yes	Daffodil International University, Dhaka, Bangladesh	Yes	Daffodil International University, Dhaka, Bangladesh (4)
2023	[49]	ICOEI	Tirunelveli, India	Yes	Centre for Development of Advanced Computing (C-DAC), Mumbai, India	Yes	C-DAC, Mumbai, India (2) Ministry of Electronics and Information Technology (MeitY), Delhi, India (1)
2023	[50]	ACM SIGCAS /SIGCHI	Cape Town, South Africa	No	Cylab-Africa/Upanzi Network Kigali, Rwanda	Yes	Cylab-Africa/Upanzi Network Kigali, Rwanda (3)
2021	[57]	ICSCCC	Jalandhar, India	Yes	NIT, Karnataka, Surathkal, Mangalore, India	Yes	NIT, Karnataka, Surathkal, Mangalore, India (2)
2018	[64]	RTEICT	Bangalore, India	Yes	B.M.S. College of Engineering, Bengaluru, India	Yes	B.M.S. College of Engineering, Bengaluru, India (1)
2020	[60]	COMS2	Gujarat, India	Yes	Sardar Patel University of Police, Jodhpur, India	Yes	Sardar Patel University of Police, Jodhpur, India (1) National Institute of Technology, Raipur, India (2)
2021	[58]	CT-RSA	Virtual Event	-	University of London, London, UK	No	University of London, London, UK (3)
2020	[61]	USENIX Security	USA	No	University of Michigan	No	University of Michigan (3)
Total			13		15		
Journal							
2021	[56]	RCS	-	-	Catholic University of Central Africa, Yaoundé, Cameroon	Yes	University of Grenoble, Grenoble Cédex 9, France (1) Catholic University of Central Africa, Yaoundé, Cameroon (1) Toulouse Business School, Toulouse, France (1)
2017	[65]	TOPS	-	-	University of Florida, FL, USA	No	University of Florida, FL, USA (5) University of Illinois at urbana-champaign, USA (1)
2019	[63]	MAT	-	-	Federal University of Technology, Minna, Nigeria	Yes	Federal University of Technology, Minna, Nigeria (2) Clemson University, Clemson, SC, USA (1)
2022	[52]	HPT	-	-	Indian Institute of Management, Shillong, India	Yes	-
2021	[55]	HPT	-	-	Indian Institute of Technology Kharagpur, West Bengal, India	Yes	Indian Institute of Technology Kharagpur, West Bengal, India (1) Manipal Academy of Higher Education, KA, India (1)

* Venue held in a developing country.
** Institution located in a developing country.

study (DFS), app analysis, app security testing, and designing & implementing solutions (D&I Solution).

User study. Security's impact on the adoption of mobile applications in developing countries is a central concern. Around 36% of publications utilize surveys and interviews to gauge this impact. Some of these studies do not exclusively focus on the security of mobile apps but explore broader factors influencing app adoption. For instance, researchers employed models such as the Unified Theory of Acceptance and Use of Technology (UTAUT), the Health Belief Model (HBM), and the Expectation-Confirmation Model (ECM) to understand the factors influencing the adoption and the continuance to adopt mobile health apps in India [52, 55]. Similarly, studies in Cameroon leveraged the UTAUT2 framework, the Technology Acceptance Model (TAM), and the Protection Motivation Theory (PMT) to examine

Table 2.7: Overview of the types of research.

Publication	User study	App analysis	DFS	App security testing	D&I solution
Albrecht et al. [58]		✗			
Al-Ameen et al. [59]	✗				
Ansong et al. [62]			✗		
Bandan et al. [54]	✗				
Bassolé et al. [34]		✗			
Castle et al. [17]	✗	✗			
Chiboora et al. [50]	✗	✗			
Chopdar et al. [52]	✗				
Ibrar et al. [33]		✗			
Kaka et al. [68]		✗			
Kamdjoug et al. [56]	✗				
Kant Kamal et al. [49]				✗	
Khatoon et al. [64]					✗
Koala et al. [32]		✗			
Kumar et al. [61]		✗			
Latifa et al. [66]		✗			
Madwanna et al. [57]		✗			
Munyendo et al. [51]	✗				
Osho et al. [35]		✗			
Prakash et al. [55]	✗				
Reave et al. [65]		✗			
Shezan et al. [67]		✗			
Sihag et al. [60]		✗			✗
Uduimoh et al. [63]		✗			
Vakare et al. [53]	✗			✗	
Number of publication	**9 (36%)**	**15 (60%)**	**1 (4%)**	**2 (8%)**	**2 (8%)**

the factors influencing users' decisions to adopt mobile banking apps [56]. The studies incorporate additional factors, including perceived security and privacy, ease of use, trust in technology, user satisfaction, and perceived usefulness. The Partial Least Squares Structural Equation Modeling (PLS-SEM) is utilized to analyze user questionnaire responses. These studies focused on specific app users and found the factors mentioned above significantly impact the adoption of mobile banking and health apps. Other user study-focused publications concentrate specifically on mobile app security. For example, authors investigated security and privacy perceptions related to cryptocurrency trading apps in India by using supervised machine learning algorithms (such as Random Forest and Logistic Regression) to analyze user reviews and sentiments [53]. In Kenya and Bangladesh, researchers explored user perceptions, behaviors, and privacy concerns associated with data collected by mobile apps [51, 59]. The studies reveal that some users recognized the necessity of data collection for app functionality, while others expressed fear or indifference.

App analysis. Most works performed in developing country contexts, approximately 60%, are dedicated to detecting vulnerabilities and privacy violations through app analysis. App analysis involves examining various aspects of a mobile app to understand its performance, security, and user behavior. A subset of the publications combines user studies with app analysis to measure the capacity of developers in implementing protection and ensuring users' security and privacy [17, 50]. However, the majority of the works have exclusively focused on app analysis, which predominantly centers on static, dynamic, forensic analysis, and a combination of static and dynamic analysis (see more details in Section 2.4.5). Despite the security challenges

mobile applications face in developing countries [15], most publications do not use specialized approaches to deal with these challenges. However, few studies have developed app analysis approaches specifically for the context of developing countries. For instance, knowing that developing countries face numerous threats in mobile financial services, Castle et al. [17] have focused on the features and development practices of mobile financial apps in this context. In other studies, authors have used approaches to specifically uncover vulnerabilities in UPI1.0 apps that banks from India used [61, 57].

Development framework study. This type of study delves into the usage of development platforms and their security guidelines in developing countries. Several studies show that Android apps often request more permissions than they need or use, impacting the security and privacy of the users' apps [69, 70]. Researchers, such as Ansong and Synaepa-Addision, think that stems from the development frameworks used [62]. Aiming to ensure better user data privacy and security in mobile banking apps, they provide in their study a case in point, exploring mobile app security by comparing different development frameworks, including native and cross-platform, used to develop mobile banking apps in Ghana. Development frameworks for developing mobile apps are not specific to the context of developing countries. Indeed, cross and native development frameworks are used worldwide. Authors found that cross-platform apps tend to use more of the requested permissions compared to native apps.

App security testing. App security analysis often involves penetration testing. It simulates attacks on the mobile app to identify and address security vulnerabilities that could be exploitable, helping in enhancing the overall security posture of the app [71]. As they feel the need to address security issues on the apps, some researchers test the security posture of mobile apps from a specific AppStore in India's mSeva AppStore) using existing automated analysis tools for several security concerns [49]. Their regressive testing methods focus on evaluating multiple facets of mobile apps before hosting in the Inda AppStore, emphasizing the critical need for robust security testing procedures. Similarly, others delve into India's mobile-based cryptocurrency trading apps' security and privacy aspects using penetration testing techniques [53]. In this work, authors use the OWASP mobile security testing framework to conduct a security analysis on the apps and identify several vulnerabilities. However, all these security testing approaches are not specific to the unique context of developing countries.

Designing & implementing solution. Proposing sophisticated techniques is pivotal in fortifying the security of mobile apps worldwide. Some authors recognized this necessity by designing and implementing specialized solutions. However, these solutions did not specifically target the concerns of developing countries. Indeed, the authors proposed a system that focuses on identifying malicious behaviors that could lead to several security concerns by analyzing system-generated logs during the execution of the app [60]. The main goal is to provide a more dynamic and insightful detection method compared to existing techniques. Besides, Khatoon and Umadevi have considered the problems facing developing country app users by creating a solution to ensure secure access, authentication, and authorization for Indian stakeholders within sensitive domains like healthcare [64]. In their study, they integrate a known secure protocol for access delegation (OAuth 2.0) and the Aadhaar identification system of India into e-Health apps to enhance their security, offering

secure access to health data and facilitating efficient interaction between patients and healthcare providers. The proposed system ensures the users' registration, login, and authentication and allows access without an Internet connection.

Answer to RQ1

Our review highlights that the research on mobile app security in developing countries is broad, ranging from user studies to the design & implementation of technical solutions. It is noteworthy, however, that most works have focused on app analysis.

2.4.3 RQ2: What are the security concerns addressed in the literature?

Table 2.8: The security concerns addressed in the literature.

Publication	Data Storage	Cryptography	Permission	Access Control	Network Comm.	IPC	3rd-party Library	WebView	Malware	Code Quality and Build	Resilience
Albrecht et al. [58]		✗	✗	✗	✗						
Ansong et al. [62]			✗								
Bassolé et al. [34]	✗		✗						✗		
Castle et al. [17]	✗		✗	✗	✗		✗				
Chiboora et al. [50]	✗	✗	✗		✗					✗	✗
Ibrar et al. [33]	✗	✗			✗	✗		✗			
Kaka et al. [68]					✗						
Kant Kamal et al. [49]	✗	✗			✗						✗
Koala et al. [32]			✗								
Kumar et al. [61]			✗	✗							
Latifa et al. [66]			✗								
Madwanna et al. [57]									✗		
Munyendo et al. [51]			✗								
Osho et al. [35]	✗										
Reave et al. [65]	✗	✗		✗	✗						
Shezan et al. [67]	✗					✗	✗	✗		✗	
Sihag et al. [60]									✗		
Uduimoh et al. [63]	✗										
Vakare et al. [53]	✗	✗			✗						
Number of publication	**10 (40%)**	**6 (24%)**	**9 (36%)**	**4 (16%)**	**8 (32%)**	**2 (8%)**	**2 (8%)**	**2 (8%)**	**3 (12%)**	**2 (8%)**	**2 (8%)**

This section outlines the key security concerns of mobile apps in developing countries, as addressed in the literature. This information highlights the importance of addressing these concerns to mitigate associated risks, including privacy data leaks, unauthorized access, remote command execution, data interception, and insecure data protection. As illustrated in Table 2.8, most of the publications have addressed security issues related to data storage, permissions, network communication, and cryptography.

Several studies reveal that apps insecurely store sensitive data in memory [63], log sensitive data in clear text or with poor encryption [17, 65], or allow data backup and enable other apps to read log files [33]. This issue can easily compromise users' security, as tools such as ADB can be used, for example, to perform complete backups of apps containing sensitive data [72].

Permissions in mobile apps help users protect restricted access, such as their information, system state, audio/camera recordings, and connecting to a paired device [73]. However, one could use them to collect users' sensitive data [51]. Studies reveal that various apps request permissions to access resources, such as writing and reading data in external storage [17, 32], and abuse these permissions, which are most categorized as risky [34, 62].

Android uses the SSL/TLS protocol to secure communication by protecting the app's data on the network. Several trusted Certificate Authorities in Android

provide certificates for many servers. In communication between a server and the app, the app should verify the trustworthiness of the server by checking the certificate [74]. Unfortunately, this is not the case in some apps. Indeed, studies reveal that several apps do not implement or properly use SSL/TLS certification validation [33, 65]. Occasionally, apps use insecure protocols, such as HTTP, for certain communication [17, 65]. Even if this communication does not contain sensitive data, it can be exploited by attackers for a man-in-the-middle attack or a phishing attack.

Studies reveal that cryptography is poorly used in several mobile apps. Some apps use custom cryptography or poorly implemented cryptographic mechanisms [65]. While others use insecure cipher modes such as ECB, outdated cryptographic protocols like MD5, SHA1, and DES, and hard-coded cryptographic keys to encrypt data [33, 50].

Besides that, some concerns are not commonly addressed in the literature. For example, there are security concerns related to code quality and resilience. Code quality addresses coding vulnerabilities originating from external sources, including app data entry points, the OS, and third-party software components [75]. For instance, Android debug mode enables the transfer of data between the app and a computer, as well as the installation and uninstallation of apps on the device, reading logcat memory, and more. The developer can enable this mode on the app for testing purposes. However, if it is not disabled before deployment, the app could allow attackers to inject code for malicious activities, access sensitive data, and so on. In some studies, researchers found that this mode is enabled in several apps [67, 50]. The resilience aspect encompasses vulnerabilities that lead to app reverse-engineering, tampering, device jailbreaking, and other similar threats. It verifies that the app operates on a trusted platform, prevents runtime tampering, and maintains the integrity of the intended app functionality [76]. Some authors have analyzed apps for resilience and found that several apps attempt to gain root access on the device, do not implement any obfuscation methods, and fail to prevent reverse-engineering [49, 50].

These concerns affect mobile apps worldwide. However, some studies addressed them by focusing on security challenges in developing countries [17].

Answer to RQ2

In most publications, investigations are related to data storage, permissions, network communication, and cryptography, concerns that are not specific to developing countries. A few studies, however, have addressed various security concerns by focusing on specific challenges in developing countries.

2.4.4 RQ3: Which apps are covered in the literature?

As shown in Table 2.9, the SLR illustrates the primary interest in financial applications, particularly mobile banking and money apps. Financial apps receive substantial attention from researchers. Some researchers analyzed the security vulnerabilities of mobile banking and payment apps within African countries, aiming to create awareness among users, businesses, and governments about potential security threats [34], to assess how these apps handle sensitive user information, and

Table 2.9: The categories of apps addressed in the literature.

Publication	Finance	Health&Fitness	Education	Communication	Travel&Local	Medical	Generic
Albrecht et al. [58]				✗			
Al-Ameen et al. [59]							✗
Ansong et al. [62]	✗						
Bandan et al. [54]	✗		✗	✗	✗	✗	
Bassolé et al. [34]	✗						
Castle et al. [17]	✗						
Chiboora et al. [50]	✗						
Chopdar et al. [52]		✗					
Ibrar et al. [33]	✗						
Kaka et al. [68]	✗						
Kamdjoug et al. [56]	✗						
Kant Kamal et al. [49]							✗
Khatoon et al. [64]		✗					
Koala et al. [32]							✗
Kumar et al. [61]	✗						
Latifa et al. [66]	✗						
Madwanna et al. [57]	✗						
Munyendo et al. [51]	✗						
Osho et al. [35]	✗						
Prakash et al. [55]		✗					
Reave et al. [65]	✗						
Shezan et al. [67]							✗
Sihag et al. [60]							✗
Uduimoh et al. [63]	✗						
Vakare et al. [53]	✗						
Number of publication	**16 (64%)**	**3 (12%)**	**1 (4%)**	**2 (8%)**	**1 (4%)**	**1 (4%)**	**5 (20%)**

to identify potentially exploitable vulnerabilities [35]. Others provide a comprehensive security analysis of the Unified Payments Interface used by some mobile payment apps in India, aiming to identify vulnerabilities and propose a more secure UPI for payment apps [57, 61]. This can be motivated by the fact that they are relatively more sensitive to security concerns compared to other categories of non-financial applications in developing countries [34]. Studies also discuss the numerous security challenges they face [15] and the types of fraud related to mobile financial services in these regions [16]. Mobile apps related to Health&Fitness receive the attention of researchers in the context of the COVID-19 pandemic through interviews and surveys to determine the factors that influence the adoption and the continuance intentions of usage of contact tracing apps [52, 55]. Aiming to provide secure access to health data and facilitate efficient interaction between patients and healthcare providers, some authors implement solutions integrating OAuth 2.0 and the Aadhaar identification system into Indian e-health apps [64]. Communication apps are studied less frequently, as well as Educational, Travel & Local, and Medical apps. Several studies do not mention the category of apps they focused on. We have grouped those apps in the Generic category.

Answer to RQ3

In the context of developing countries, researchers primarily focus on financial mobile apps, including mobile banking and mobile money applications.

Table 2.10: Summary of the techniques used to detect security issues.

Publication	Static	Dynamic	Hybrid	Forensic
Albrecht et al. [58]		✗		
Ansong et al. [62]	✗			
Bassolé et al. [34]			✗	
Castle et al. [17]	✗			
Chiboora et al. [50]			✗	
Ibrar et al. [33]	✗			
Kaka et al. [68]	✗			
Kant Kamal et al. [49]			✗	
Koala et al. [32]	✗			
Kumar et al. [61]			✗	
Latifa et al. [66]	✗			
Madwanna et al. [57]			✗	
Osho et al. [35]				✗
Reave et al. [65]	✗			
Shezan et al. [67]			✗	
Sihag et al. [60]		✗		
Uduimoh et al. [63]				✗
Vakare et al. [53]			✗	
Number of publication	**7 (28%)**	**2 (8%)**	**7 (28%)**	**2 (8%)**

2.4.5 RQ4: What techniques are used to detect security issues?

This section examines the techniques used to identify security concerns in mobile apps in developing countries. Numerous methods exist for identifying vulnerabilities in these apps, with static and dynamic analysis being the most prevalent. Static analysis involves examining software without executing it, while dynamic analysis involves examining software during runtime. These two techniques are applied based on specific needs. To enhance analysis efficiency, researchers frequently integrate static and dynamic analysis to create a hybrid approach.

As illustrated in Table 2.10, the literature indicates a predominant reliance on static and hybrid analysis to address app security issues in developing countries.

Additionally, dynamic analysis is occasionally used, along with forensic analysis, which involves examining an app's traces in the device's memory.

The results show that most publications have only used off-the-shelf techniques and tools to uncover security issues [33, 34, 35, 49, 50]. Others have combined those tools with specialized techniques to identify features that lead to data leakage [17], analyze communication flows between servers and mobile apps [17, 65], and examine registration and transaction processes [57, 61].

Answer to RQ4

Researchers predominantly have employed static and hybrid analysis methodologies to scrutinize mobile apps for potential security issues. Nevertheless, most did not specifically adapt their works to the specific concerns of developing countries.

2.4.6 RQ5: What are the specific characteristics of malware and the techniques used to compromise devices?

The literature review revealed only three publications, i.e., [34, 57, 60], that identified malicious apps through the exploration and analysis of mobile applications. In these publications, authors found mobile trojans to be the most prevalent type of mobile malware among the apps they studied. More specifically, in the first identified paper, Bassolé et al. [34] explored malware detection and identified five malware instances in a set of 53 mobile banking apps, including three trojans, using VirusTotal. In the second paper, Madwanna et al. [57] develop a Trojan to assess the security level of an app that leverages the Indian payment system, Unified Payments Interface (UPI). The Trojan tries to exploit vulnerabilities present in UPI. More specifically, the Trojan disables client-side security and creates a command and control (C&C) server. The authors demonstrate that the malicious app can collect users' phone numbers from all phones on which it is installed and send them to the attacker via SMS. Finally, in the third paper, Sihag et al. [60] identified several malicious apps when applying their dynamic analysis approach to a set of apps, including those that attempted to jailbreak the phone, root the device, and use superuser commands. Their approach focused on detecting malicious behaviors by collecting and inspecting system-generated logs.

These studies mainly focused on apps that are used in the financial domain [34, 57] and in other categories of apps such as social networks, games, productivity, messaging, etc. [60]. However, the studies did not mention the information assets that are attacked the most and the attack vectors.

Answer to RQ5

The literature revealed very few studies on detecting mobile malware in developing countries. These studies primarily identified malware, including trojans, jailbroken apps, and apps that attempt to root devices or execute superuser commands. These malware target mainly app categories, such as finance, social networks, games, productivity, and messaging. There are no specific details on the payloads used, the threat actors involved, the attack vectors, or the most targeted information assets.

2.4.7 RQ6: What motivated researchers to investigate mobile app security in developing countries?

With Smartphones becoming increasingly popular in developing countries[6], a surge in mobile app usage has raised awareness about security issues among researchers. Researchers motivate their studies by mentioning the growing concerns regarding mobile app security that impact many users in developing countries. They aim to encourage practitioners to enhance user protection and security measures, especially in the case of mobile financial services. The dearth of studies on security issues in developing countries inspires researchers to fill this gap. Furthermore, examining the relatively low adoption rates of certain mobile apps also fuels researchers' curiosity.

[6]Source: `http://www.osiris.sn/En-Afrique-subsaharienne-le-taux-d.html`

This curiosity prompted investigations into whether mobile app security plays a role in influencing adoption patterns.

In certain instances, researchers strive to reassure users, mitigate the threat posed by malicious apps, and address the unique security challenges prevalent in their respective regions. This multifaceted approach reflects a commitment to understanding, confronting, and ultimately improving the landscape of mobile app security in the context of developing countries.

Answer to RQ6

Researchers generally motivate their papers by citing the increased adoption of smartphones, coupled with a significant rise in mobile app usage, growing concerns surrounding mobile apps, and the lack of studies in developing countries.

2.5 Discussion

This section discusses various aspects of mobile app security in developing countries. We address unexplored research directions, emerging trends, and issues related to vulnerabilities, outline future challenges, and present the threat to the validity of this work.

2.5.1 Researches performed and needs

In the domain of mobile app security in developing countries, several uncharted research directions hold the potential to drive innovation and enhance security practices. Certain avenues have been much more explored.

2.5.1.1 Explored Research directions.

The identified papers investigate a few relevant research avenues that could contribute to the development of more secure apps.

Several research papers have conducted user studies through interviews and surveys, which offer valuable insights into user behavior and perceptions when using mobile apps, as well as their intentions for using specific mobile apps. This type of contribution provides a deep understanding of user experiences and sentiments. It also guides the development of more secure apps, as it has allowed researchers to provide actionable recommendations and guidelines for developers, policymakers, and app markets. Researchers employ advanced methods and frameworks, including machine learning algorithms, UTAUT, and PLS-SEM, to provide robust analysis. However, we observed that they often use a limited sample size. The sample size used could be representative, taking into account gender diversity and geographic area, to provide stronger results. We have observed that researchers primarily focus on the adoption of financial and healthcare apps. This is understandable since they are two crucial domains. Nevertheless, the scope could be extended to cover other domains.

Researchers have primarily focused on app analysis approaches, including static, dynamic, hybrid, and forensic analysis. Their work is primarily based on using off-the-shelf automated security detection tools, among which some have limited capabilities to detect specific vulnerabilities [33]. Each study focuses on specific

security aspects according to the category of apps it addresses. We have noticed that financial apps have received more attention from researchers than other categories of apps. This attention is not driven by the fact that financial apps are the most widely used in developing countries; therefore, researchers should also attach importance to mobile apps in other domains, including healthcare, gaming, and social media, as the security of these apps could affect millions of users.

Researchers have also minimally addressed app security testing [49, 53]. This type of study is useful since it not only allows the identification of vulnerabilities statically but also provides a runtime analysis of vulnerabilities, raising awareness on the user and developer sides. According to the threats targeting mobile apps in developing countries [25, 15, 16], there is a pressing need for specialized solutions to deal with these problems. Unfortunately, only a single study has been provided proposing a context-specific tool to ensure secure access, authentication, and authorization within the healthcare domain in India [64]. Another work has proposed a system that collects and evaluates behavioral activity for malicious intent from targeted apps [60]. However, these approaches are not specific to developing countries. The existing literature has not mentioned any security testing tools specifically tailored for mobile apps in developing countries. Researchers typically use pre-existing open-source tools developed for a global context, even for app analysis, rather than creating region-specific solutions.

We have noticed a significant lack of research into the security of mobile applications in developing countries. More studies should be proposed, ranging from considering more app categories to proposing solutions adapted to developing countries. Many approaches exist in the literature worldwide. However, applying them in the developing country context can lead to missing interesting findings because the types of functionality and errors in mobile apps could be different.

2.5.1.2 Unexplored Research directions.

Additionally, there is a domain that warrants further exploration. Mobile malware detection warrants further exploration. Indeed, mobile phone users worldwide are increasingly falling victim to a rising tide of mobile malware infections [77]. These infections disproportionately impact users in developing countries. According to Carter [26], the top ten countries most severely affected by mobile malware are developing countries, meaning that they deserve important attention from researchers. Despite this, we have found only two studies focusing on mobile malware detection in these regions. The first one designed a system that analyzes real-time app interactions for malicious behavior [60], while the second one conducted a vulnerability analysis using VirusTotal for malware detection [34]. While several advanced malware detection methods exist, utilizing AI techniques such as machine learning and deep learning, these studies primarily employed dynamic analysis techniques to address mobile malware [60]. Apart from these two works, no other studies have delved into the critical issue of mobile malware detection. It is noteworthy that researchers tend to focus on analyzing mobile apps for security vulnerabilities rather than proactively addressing the issue of mobile malware detection despite the escalating threat posed by mobile malware in developing countries.

Another research direction could be to explore proof-of-concept, allowing the exploitation of vulnerabilities found in mobile apps. This can help to improve its security. In our study, we have identified only one paper in which authors have

outlined hypothetical attacks on mobile money apps without demonstrating how these attacks could be executed [17]. This hesitancy may be attributed to the constraints inherent in conducting proof-of-concept experiments. In fact, various factors, such as governmental regulations, resource constraints, or the requisite access to specific accounts, can hinder the practical exploitation of vulnerabilities. However, presenting straightforward use cases can potentially address these issues. Despite these challenges, investigating how adversaries can exploit vulnerabilities remains crucial for improving mobile app security in developing countries.

2.5.2 Research trends

We were surprised by the few papers we have identified in this SLR. Indeed, we have identified twenty-five publications, and the first appeared in 2016 [17]. In general, there is an average of three relevant publications annually until 2023, indicating the scarcity of research in this area. To resolve this problem, more comprehensive work is required to address the numerous security issues in the rapidly expanding mobile app landscape in developing countries. In Section 2.5.4, we proposed several future types of research identified when analyzing publications to help researchers fill the gap.

2.5.3 Comparison with existing secondary studies

In the literature, we have identified several secondary studies, each with a different focus compared to our SLR. For instance, a state-of-the-art survey on mobile banking analyzed and classified the challenges and security issues of mobile banking services in Uzbekistan [41]. This work focused on the Wireless Application Protocol (WAP), including its use for data encryption, authentication processes, SMS banking, and protection against virus attacks. Similarly, another survey focused on identifying the issues and challenges faced by customers using mobile banking services [43]. Additionally, researchers conducted a literature review on mobile government (m-government), analyzing studies that presented the challenges, benefits, and success factors for implementing m-government [42]. The similarities between these studies and our study are that they investigated the security problems faced by mobile banking and mobile government in developing countries. Some of these security problems are linked to network communication and cryptography, security concerns that we also identified in RQ2 of our study. However, they did not primarily focus on mobile applications as our study does.

Other studies have examined mobile apps in developing countries and various other areas. For example, reviews of mobile health apps (mHealth) have explored the application of recommendations and security and privacy regulations [27, 30], the current state of the mHealth market [39], and the security challenges of mobile educational apps [31]. Each of these studies focused on specific categories of apps. The main similarities we noticed in these studies are that they (1) focused on the categories of apps we considered, i.e., mHealth and educational apps, and (2) explored security concerns related to cryptography, access control, network communication, and data storage [30].

In addition to these specific apps' categories, our SLR considers security problems across all categories of mobile apps. We categorize the publications by types of contributions to highlight future research directions. Furthermore, we explore the types of issues identified in the apps and the methods researchers used to identify

these issues.

2.5.4 Future challenges

Our discussion highlights perspectives from the existing literature and presents new challenges that need to be addressed to enhance mobile app security in developing countries.

2.5.4.1 Challenges from the literature.

The authors of the papers listed in our SLR have identified several open challenges to extend research or explore new directions. The identified perspectives include:

1. *Investigating development frameworks.* The authors explored some app development frameworks to understand the impacts they have on the security of the apps. They argue that certain frameworks allow the development of more secure apps than others. The study focuses on two development frameworks, and the authors suggested extending it by considering more frameworks.

2. *Reducing the requested permissions.* Some studies investigated mobile app security by focusing on permissions. The authors found that some apps abuse users by requesting unnecessary permissions, which could compromise their security. These authors suggest investigating and proposing solutions to reduce the permissions requested by apps to protect user privacy.

3. *Developing alert tools.* There are also perspectives to develop tools to alert users and administrators about potentially dangerous permissions used by apps.

4. *Expanding studies.* When performing interviews and surveys, researchers used a small number of participants and focused on specific areas and institutions. To obtain more comprehensive results, they suggested expanding the studies to cover diverse contexts, institutions, and participants.

5. *Focusing on specific aspects.* To refine the analysis results, researchers suggested carrying out studies that focus on specific aspects of financial apps, such as password reset procedures and accountless money transfers.

2.5.4.2 Future research directions.

Based on the insufficiently explored research directions mentioned above and the need for more specific approaches, we have identified, in addition to the challenges in the literature, the following future research directions:

1. Investigate unexplored research areas in mobile app security analysis, targeting the specific challenges of developing countries.

2. Extend applications of state-of-the-art research results to explore security concerns beyond fintech apps to other critical categories, such as health and education apps.

3. Investigate malware targeting developing countries. In particular, the types of malware used, the payloads of these malware, the techniques and procedures used by attackers to compromise devices in these regions, the active threat

actors, and the information assets that are attacked the most should also be explored. Furthermore, analysis techniques should be developed that are suited for (1) pre-installed apps in low-cost devices and (2) available and still-in-use insecure technologies (e.g., USSD).

Mobile app security in developing countries is an evolving field with numerous opportunities for research and improvement. Addressing these research directions and challenges is essential to enhance the security of mobile apps and protect users in these regions.

2.5.5 Threat to Validity

Our primary objective was to assemble publications discussing mobile app security in developing countries. However, there are potential threats to the validity of this study.

Internal validity. We established exclusion criteria to eliminate irrelevant publications during the selection process. Notably, we excluded books and thesis reports, as we deemed them less relevant, assuming that pertinent information would likely be found in conference or journal papers. The authors reached a consensus to remove certain publications after reviewing them and determining that they were irrelevant to our study. We also modified our search keywords to capture more relevant publications than in our initial attempt. However, this adjustment may have led to the omission of some pertinent publications.

External validity. Our study did not include works published after 2024, as we completed our data collection prior to this date. Consequently, publications released after our cutoff date were not considered, which means that the results and arguments presented in this study may not be applicable to these newer publications.

Construct validity. To streamline our search on the Springer digital library, we performed a general search and applied filters to select publications within the field of Computer Science. We then narrowed our focus to conference papers, articles, and papers written in the English language. Since some developing countries have French as an official language rather than English, we may have missed relevant publications written in French. Additionally, we used a Python script to refine the results by targeting the titles and abstracts of the collected publications. This script constructs a search string based on a list of keywords and searches for it in the titles and abstracts. However, this method may inadvertently exclude some relevant publications. We found no irrelevant publications when verifying the results we obtained after running the script. All publications we obtained contain our search string; thus, no false positives have been found. However, we did not exclude the possibility of having false negatives, that is, excluding relevant publications. Our script is publicly available to the research community in our replication package [36] for transparency and verification. Furthermore, we excluded some publications when assessing their quality. This exclusion was not because the publications did not investigate mobile app security in developing countries. We excluded them because they did not meet our quality criteria, which could potentially include biases in our dataset.

While these actions were intended to refine our results, they may have inadvertently excluded some potentially relevant publications. Consequently, we acknowledge the possibility of biases in our study. This means we might have missed publications that should have been included or excluded. However, during our data analysis, we

did not identify any additional publications to exclude, as each one helped us address at least one of our research questions.

2.6 Summary

In this chapter, we delved into mobile app security in developing countries. We meticulously examined 25 publications from various conference and journal venues throughout this process. Our goal was to shed light on the diverse research directions, security concerns, categories of apps addressed, investigation techniques, and researcher motivations in the domain of mobile app security within developing countries. Our key findings from this SLR include ① a lack of studies, ② limited research directions, ③ a focus on financial apps, and ④ a lack of specific analyses.

Taken together, these findings point out the importance of expanding and diversifying mobile app security research in developing countries. By identifying these gaps and trends, this SLR provides a foundation for future work that can better align with local realities and global security goals. It contributes to both academic research and practical application by mapping the current landscape, highlighting critical gaps, and proposing directions for future research. It serves as a reference for researchers, developers, and policy makers aiming to improve the security of mobile apps in these regions, supporting more secure and inclusive digital ecosystems.

It is essential to acknowledge that this study is limited to developing countries, excluding emerging ones. Future studies on SLRs may expand on this work by including emerging countries, thereby providing a more comprehensive perspective on mobile app security in diverse socio-economic contexts.

3

On the security of pre-installed Android apps in low-cost devices

In this chapter, we present PiPLAnD, a pipeline that extracts APKs from physical devices and applies static analysis to detect sensitive-data leaks, manifest misconfigurations, and suspicious behaviors in pre-installed apps. Using PiPLAnD, we analyzed 2050 *pre-installed APKs collected from nine devices (Infinix, itel, Tecno). Our findings show that 358 apps (17%) leak sensitive information, 380 apps (19%) export sensitive components without adequate protection, and numerous apps exhibit risky behaviors (310 execute dangerous commands, 121 access/send/delete SMS, 53 perform silent installation actions, 14 can collect log data, 512 read clipboard data, and 24 use malicious URLs). We also identified a vendor-shipped package that appears to exfiltrate device identifiers and location to a third-party vendor. These results indicate that pre-installed software on widely distributed, low-cost devices can pose real privacy and security risks to end users.*

This chapter is based on the work published in the following research paper [78]:

- Diallo, A., Diop, A., Kaboré, A. K., Pilgun, A., Samhi, J., Bissyandé, T.F., and Klein, J., On the security of pre-installed Android apps in low-cost devices. 17th EAI International Conference on Africa Internet Infrastructure and Services (EAI AFRICOMM) (2025).

Contents

3.1 Overview

The Android operating system has enabled affordable smartphones for millions of people worldwide. Devices typically ship with manufacturer- or vendor-installed systems and third-party apps, which can become a distribution vector for malware and privacy-invasive functionality, especially in developing regions [6]. Low-cost smartphones remain widely used across Africa [7] and have helped reduce the digital divide by expanding access to services [8, 9]. For example, roughly 20–22 million people in South Africa use smartphones, accounting for one third of the population [7]. Feature phones and inexpensive devices are still popular in Africa, which preserves the market opportunity for the expansion of low-cost Android devices. Although low-cost Android initiatives have improved device accessibility [11], the pre-installation process can be abused to embed harmful apps that reach large user populations.

Multiple reports document dangerous apps shipped with devices [79, 80, 81, 82, 83, 84]. These apps have been observed exfiltrating personal data to remote servers [79], harvesting financial credentials via clipboard monitoring [80], performing silent installations [83], or enrolling users in paid services without consent [84]. Several of the reported apps appear specifically on low-cost devices, including phones sold in Africa [79, 84].

A few studies collected firmware images from vendor websites and online forums, extracting pre-installed apps for analysis [83, 85, 86, 87]. The extracted apps are then examined for suspicious permissions, misconfigurations in manifest files [83, 86, 87], malware [83], and data leaks [85, 88, 89]. Other studies have directly extracted pre-installed apps from physical devices [88, 89], allowing for a more comprehensive assessment across various Android brands. Yet, research on pre-installed applications remains limited since Android device manufacturers are expected to strictly follow compliance guidelines enforced by Google [90]. Unlike globally known brands, Android devices sold in African markets are often very low-priced. This raises concerns about the trade-offs manufacturers make between cost and security.

We studied nine low-cost device models (Infinix, Tecno, itel) – all produced by TRANSSION and together accounting for 51% of the African smartphone market in 2023 [91]. We extracted a dataset of 2050 pre-installed APKs for analysis. Pre-installed apps on these devices have received little systematic analysis despite their prevalence. The most notable work is by Elsabagh et al. [85] who examined pre-installed apps from Infinix and Tecno, focusing on privilege escalation. To the best of our knowledge, no prior work has systematically analyzed pre-installed apps across these low-cost Android devices commonly sold in Africa. To explore this gap, we developed *PiPLAnD*, a pipeline that extracts APKs from physical devices and applies taint-based leak detection, pattern-driven behavior scanning, and manifest/component inspection to find data leaks, insecurely exported components, and suspicious behaviors. Our work builds upon existing work by taking a more in-depth approach. For instance, previous studies identified exported components by analyzing only the application manifests. In contrast, we extended this analysis by examining the corresponding code to detect components that provide access to sensitive data. Moreover, while prior research on data leaks has mainly focused on those occurring through Android or Java API methods, our analysis also considers leaks introduced via third-party libraries.

Our analysis uncovered multiple suspicious pre-installed apps across the tested

devices. A substantial fraction of these apps leak sensitive information (for example, IMEI, IMSI, location) and exhibit manifest misconfigurations that expose components without adequate protection. In a notable case, *PiPLAnD* identified the package *com.transsion.statisticalsales*, which was not flagged by VirusTotal. These findings underscore the need for independent audits of pre-installed software on low-cost devices in developing regions.

Below we summarize our main contributions:

① We present *PiPLAnD*, a pipeline to systematically inspect pre-installed apps from Android devices.
② We collected a dataset of 2050 pre-installed APKs from nine low-cost device models (Infinix, Tecno, itel).
③ We identify manifest misconfigurations: 380 app versions (19%) export sensitive components without adequate protection.
④ We quantify sensitive-data leakage: 358 apps (17%) leak identifiers or location data.
⑤ We document widespread suspicious behaviors in pre-installed apps.
⑥ We identify 24 pre-installed apps (around 1.2%) using malicious URls.

3.2 Background and Related Works

3.2.1 Background

This short background defines terms used in the paper and clarifies our measurement scope.

Android firmware and pre-installed apps. Firmware boots device hardware and the Android OS[1]. Devices ship with *system apps* and *manufacturer-supplied* pre-installed apps [92]. *System apps* can run with elevated privileges or reside in privileged locations (e.g., `/system/priv-app`). Some devices in our dataset use Android Go, a lightweight Android configuration for entry-level hardware. Android Go was designed to optimize its running on low-end devices; however, the security implications of this optimization are not fully understood [10].

Exported components and misconfigurations. Apps declare components in AndroidManifest.xml. A component is "exported" if other apps can start or bind to it (via `android:exported` or an `intent-filter`). Exported components without proper permissions or access checks form an attack surface and are treated here as security misconfigurations [93].

PII, sources, sinks, and leaks. PII includes identifiers and user data (IMEI, IMSI, phone number, location) [94]. A SOURCE is a method that reads sensitive data (e.g., `getLastKnownLocation()`); a SINK is a method that can exfiltrate data (network send, SMS, world-readable storage). When an app allows to get this data and send it outside the app itself, in this case, we talk about data leakage (or data leak). Static taint analysis tools like FlowDroid are well known to detect leaks in Android apps [95].

Low-cost device. In this study, we often use the terms "low-cost", "low-end", "cheap", and "affordable" when talking about devices that are not expensive but affordable, specifically targeting devices primarily used in Africa.

[1] `https://www.androidpolice.com/what-is-firmware/`

3.2.2 Related Works

To the best of our knowledge, this work is the first one that focuses on analyzing Android apps pre-installed on the devices primarily used in Africa. However, there are some existing works about the analysis of pre-installed apps in Android devices.

Indeed, Zheng et al. [83] presented a tool named DroidRay that extracts statically and dynamically pre-installed apps from 250 firmware downloaded from forums and websites. The static extraction consists of extracting pre-installed apps directly from the firmware images. They also flashed the image into a device and then dynamically extracted the pre-installed apps using ADB commands. DroidRay performs pre-installed app analysis and system analysis by extracting the "SharedUserId" attribute from the AndroidManifest.xml and the signature information from the RSA file before comparing them with the default signatures they found from the AOSP. They also analyzed the apps in VirusTotal to detect potential malware. It performs static and dynamic analysis of the Android firmware by doing a system signature vulnerability detection, a network security analysis, and a privilege escalation vulnerability detection. In our proposed pipeline, we leveraged the technique of dynamic extraction to directly extract pre-installed apps from a physical device using ADB commands rather than collecting firmware and flashing it into a device.

Mitchell et al. [96] designed DexDiff, a system for assessing the security impacts of vendor customization to the official Android system. DexDiff helps the security analyst, who first retrieves the pre-installed apps and libraries from the phone and then builds their corresponding base binaries from the release branch in AOSP on which the phone is based. DexDiff compares each pair of these binaries obtained and evaluates the security impacts of individual modifications. The only similarity between this study and our approach is that it extracts pre-installed apps from the phone. However, the proposed tool did not automate this process. The apps are supposed to be extracted before using DexDiff. Contrary to our approach, PiPLAnD automated the process of extraction and analysis.

Elsabagh et al. [85] proposed a static analysis tool named FIRMSCOPE to identify unwanted functionality in pre-installed apps by analyzing the Android firmware. FIRMSCOPE extracts pre-installed apps from the firmware and then performs a taint analysis with context-sensitive, flow-sensitive, field-sensitive, and partially object-sensitive. Specifically, it focuses exclusively on identifying the increase in privileges.

Gamba et al. [88] presented a large-scale study of Android pre-installed using crowd-sourcing methods. In this study, the authors built an Android app, Firmware Scanner, that looks for and extracts pre-installed apps when installed on a device. This study performs permission analysis using Androguard, static analysis leveraging existing tools such as Androwarn, FlowDroid, and Amandroid, as well as apktool and Androguard frameworks to identify unwanted behaviors, and traffic analysis using the crowd-sourced Lumen mobile traffic dataset to see app real-world behaviors. Compared to this work, we did the same by extracting pre-installed apps from the physical device, but using ADB commands rather than an installed app. We also performed a taint analysis using FlowDroid, but by considering third-party libraries as well.

Blázquez et al. [89] proposed FOTA (Firmware-Over-The-Air) Finder to automatically classify a given APK as FOTA or not based on Androguard, using the dataset of pre-installed apps from Firmware Scanner [88]. Then, they performed

behavior analysis relying on FlowDroid and Amandroid for a taint analysis and a modification of Androwarn to analyze the use of API calls. This part of the work is a bit similar to our data leak detection using FlowDroid. However, we considered all the pre-installed apps extracted from devices, and also performed malware detection.

Hou et al. [86] performed a study in which they collected firmware images from vendors, official websites, and open source repositories, and CVE data to link them with pre-installed apps. In this study, they proposed a tool named AndScanner that automates the extraction of pre-installed apps from firmware images before analyzing them. AndScanner proposes an analysis of the security patches of the firmware to know if it has been patched in time and if the security issues have been fixed. It also performs app analysis by analyzing the pre-installed apps using Androguard to identify misconfiguration in the manifest file and CryptoGuard to detect cryptography misuse. Our study is different since we did not only focus on analyzing the manifest files, but also analyzed the cryptography misuse. However, we leveraged on Androguard framework in our pipeline.

More recently, Sutter and Tellenbach [87] proposed FirmwareDroid, an automated static analysis tool for pre-installed apps. This tool automates the process of extracting pre-installed apps from firmware images and their analysis using existing tools, including Androgurad and Exodus. The study identifies the advertising tracker libraries used with Exodus and the permissions pre-installed apps inherited with Androguard. The authors have integrated 8 open source static analysis tools in FirmwareDroid, which could be used for further analysis.

Almost all of these studies are a bit similar since they follow almost the same approach, such as collecting firmware from the Internet, extracting the pre-installed apps, and analyzing them. Our approach allows the extraction of pre-installed apps from physical devices, the detection of apps having suspicious behaviors, the detection of data leakage, and security misconfiguration on the Manifest files. Furthermore, our approach is particularly tested on low-cost devices sold in Africa. However, with this approach, every brand and every Android device can be inspected and analyzed. Consequently, we do not have a limitation related to missing some brands or firmware.

3.3 Threat model

In this work, we analyze pre-installed applications on low-cost Android devices distributed across several regions in Africa. Our threat model focuses on identifying the risks posed by such applications to end-user security and privacy. We consider several potential adversaries in this context. Device manufacturers and firmware vendors may embed apps that harvest user data for commercial purposes, engage in silent app installation for monetization, or include malicious apps for remote control. These actors often have full control over the firmware and can sign system apps with the platform certificate, enabling them to request and use powerful system permissions. Third-party app developers whose apps are bundled with the firmware may leverage this privilege to collect data or bypass runtime permission requests, knowing the user cannot uninstall them. These actors may also exploit insecure IPC endpoints exposed by other system apps. Malicious apps may attempt to interact with vulnerable or exported components in system apps. They may also trigger operations such as sending SMS messages, accessing private content providers, or silently installing other apps without user interaction. Remote attackers may exploit

vulnerabilities in native libraries or insecurely exposed APIs to gain remote access to sensitive data or execute arbitrary code on the device. The primary targets for these adversaries include sensitive user data, system-level capabilities, and operating system logs.

Several architectural and implementation weaknesses contribute to the attack surface in these devices.

Exported components. Many system apps expose activities, services, broadcast receivers, or content providers without proper permission checks, making them accessible to third-party apps.

Use of shared UIDs and platform certificates. Apps signed with the same platform key or sharing the same sharedUserId can bypass Android's app sandbox model, leading to unexpected privilege sharing.

Access to privileged permissions. Pre-installed apps may request and be granted system-level permissions such as *READ_LOGS*, *INSTALL_PACKAGES*, *SEND_SMS*, or *RECEIVE_BOOT_COMPLETED*, usually unavailable to user-installed apps.

Insecure inter-process communication (IPC). Some apps use implicit intents or lack permission enforcement for critical operations, making them susceptible to misuse.

Dynamic code loading and reflection. These mechanisms allow apps to bypass static analysis and may be used to execute code not bundled with the APK, complicating detection and increasing the risk of malicious behavior.

Based on the above attack surfaces, we consider several concrete threat scenarios: ① A pre-installed system app silently installs other applications from a remote server using PackageInstaller APIs, bypassing user consent. This may be used to deploy adware, surveillance tools, or additional bloatware. ② An app signed with the platform key exposes a service that allows sending SMS messages. A third-party app exploits this service to send messages to premium-rate numbers, leading to financial loss for the user. ③ A system app collects logcat output and writes it to a file readable by any app on the device, thus leaking sensitive data. ④ An exported content provider grants read access to personal data such as SMS messages or contacts without requiring any permission, allowing any app on the device to exfiltrate user information.

Our threat model assumes that the end users are not root users and cannot remove or inspect system apps without significant technical effort. Devices under study are not regularly updated, making it likely that outdated apps and libraries persist in the firmware. We also assume that adversaries can exploit vulnerabilities in system apps using either locally installed apps or remote payloads (e.g., via malformed media files or silent network communication).

3.4 Low-cost Android devices in Africa

Upon careful investigation of the African mobile device market, we identified three popular brands shipping Android devices under 100 US dollars price – Infinix, itel, and Tecno. All these devices run Android Go Edition. We acquired 7 devices in total from these brands for our study.

3.4.1 Android Go Edition

Android Go Edition is a lightweight configuration of standard Android designed for entry-level devices with limited memory ($\leq 2\,\text{GB}$) and storage [10]. Android Go ships with resource-optimized "Go" versions of Google apps, but users can still install apps from the Play Store. Android Go may receive updates less frequently than standard Android [97, 10]. To reduce resource consumption, Android Go disables several features by default [10]:

- Picture-in-picture support;
- `SYSTEM_ALERT_WINDOW` permission (display over other apps);
- Split-screen / multi-window;
- Live wallpapers;
- Multi-display;
- Launcher shortcuts (deep shortcuts);
- Reduced maximum width/height for images in remote views;
- VR mode.

Android Go is not necessarily less secure than standard Android. In fact, users may gain security benefits from the removal of the `SYSTEM_ALERT_WINDOW` permission, which has often been abused by malicious apps [98]. We did not find literature that evaluates the security implications of the other optimizations introduced in Android Go. However, Android Go devices typically receive fewer updates and have a shorter support lifecycle, which can leave users without critical security patches [99].

Many pre-installed apps on some device brands are not available on Google Play. They may come from unverified third-party vendors that may potentially distribute malicious apps to the end users.

3.4.2 Devices and Pre-installed apps

From our nine devices (Infinix, itel, and Tecno) we extracted 2050 pre-installed APK files, including both system and third-party apps, which form the dataset used in this study. Most of the pre-installed apps extracted cannot be found in the Google Play Store, as shown in Figure 3.1. Several brands provide their own app stores. In particular, the Palm Store is reported to be the official app distribution platform for Infinix, Tecno, and itel [100]. Palm Store is preinstalled on the devices we examined and allows users to install, uninstall, and update apps. According to the provider, the store performs automated and manual security checks, compatibility testing, and content-compliance monitoring [100].

System apps are signed by multiple different certificate authorities, and the dominant signer varies by device. For example, Table 3.1 shows that on the Infinix SMART8, 68% of system apps are signed by Infinix; 20% by Google; 4% by Transsion; 2% by Tecno; 1% by Facebook; 1% with the default AOSP certificate; and 4% by other authorities. System apps are usually found on */system/app* and */system/priv-app* folders. However, when extracting the APK files, we have found that the apps have been distributed not only in these two folders, but in several others as well. Figure 3.2 shows an example of the folders on Infinix. In Table 3.2, we present a possible classification of the pre-installed apps by origin based on their paths. We describe each path presented in this table as follows [2]:

- **/system/app/** and **/system/priv-app/** - They are the traditional locations for system and privileged apps. Apps in **/system/priv-app/** get signature-level or privileged permissions (e.g., ContactsProvider, Settings, Telephony).

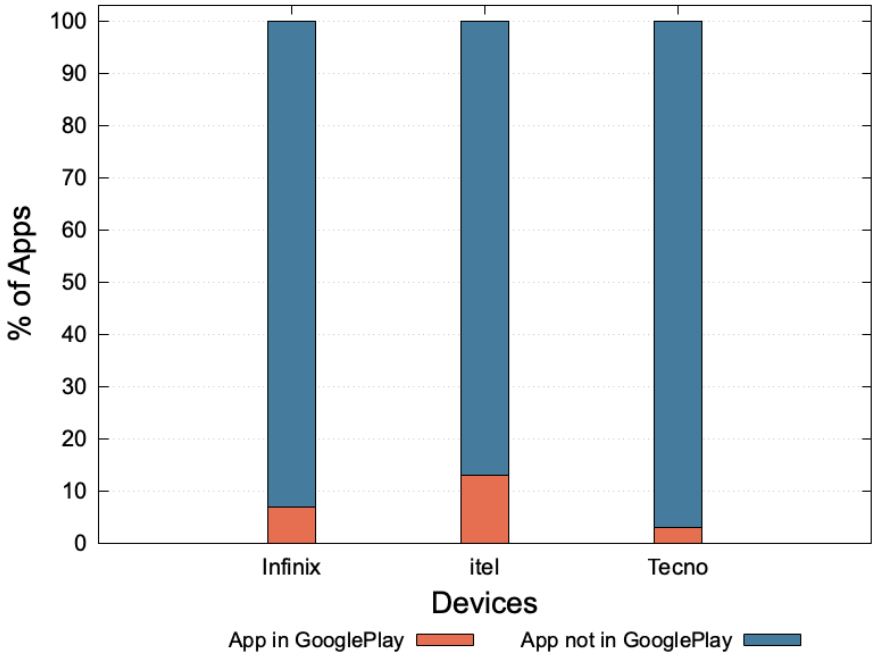


Figure 3.1: App present in Google Play vs. App not present in Google Play

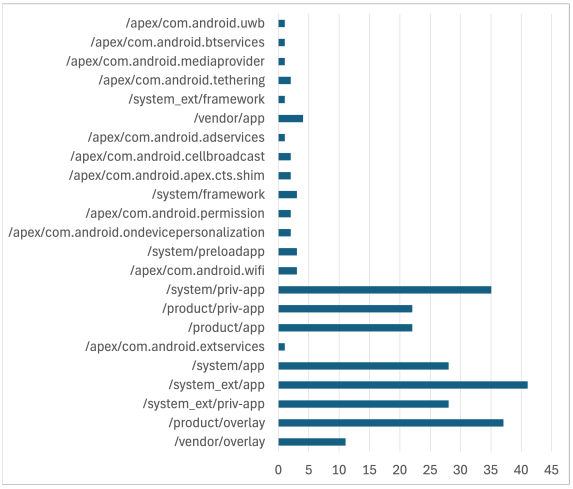


Figure 3.2: Location folders of the system apps in Infinix

Table 3.1: System apps grouped by certificate authority for some devices.

Signed by	***Infinix SMART8***	***itel A50***	***Tecno POP8***
Infinix	68%	0%	0%
itel	0%	71%	0%
Tecno	2%	1%	68%
Transsion	4%	2%	3%
Google	20%	22%	21%
Facebook	1%	1%	2%
Default	1%	1%	1%
SW	0%	0%	2%
Others	4%	2%	3%

Table 3.2: Classification of pre-installed apps by origin using their paths.

Partition	**Owner**	**Typical App Type**
/system/app, /system/priv-app	Google / AOSP	Core OS apps
/apex/	Google	Modularized system components
/system_ext/	OEM	OEM system extensions
/product/	OEM / ODM	Device-specific apps
/vendor/	SoC Vendor	Hardware-dependent apps
/system/preload/	OEM / Partner	Third-party preloads
/system/framework/	AOSP / OEM	Framework-level components

These folders are still widely used but less populated on Android 10+ because many components have moved elsewhere.

- **/apex/** [101] - Introduced in Android 10, the Android Pony EXpress (APEX) allows modular updates of critical components via Google Play. It holds APEX modules that behave like mini system partitions. Each APEX module provides a critical system component (e.g., ART runtime, media codecs, time zone data). Some APEX modules include APKs that resemble apps but are actually essential services.
- **/system_ext/** - Introduced in Android 11, it holds extensions to core system components without polluting **/system** and is used when OEMs need to add privileged services, frameworks, or system apps that rely on system internals.
- **/product/** - It is a partition created to separate OEM modifications from the core /system partition. Many OEMs place non-critical system apps here so they can update /system independently. It contains product-specific system properties (/product/build.prop), product-specific RROs (/product/overlay/*.apk), product-specific apps (/product/app/*.apk), product-specific priv-apps (/product/priv-app/*.apk), and more.
- **/vendor/** - It is one of the core vendor partitions. It contains hardware abstraction layer (HAL) implementations, drivers, and vendor-provided system apps or daemons. It also occasionally includes pre-installed hardware utility apps (e.g., camera service, modem loggers, battery manager). Vendor apps usually depend on vendor-specific libraries or permissions.

- **/system/preload/** - This folder often contains region- or carrier-specific preloaded apps. These apps can be installed at first boot (using package manager preinstall script) or mounted as read-only preloaded APKs that can't be uninstalled easily. It is common on low-cost devices. Preloads may not show as system apps until after the first boot, because the Package Manager installs them into **/data/app/**.
- **/system/framework/** - It contains core Android framework JARs and occasionally pre-optimized system apps or shared code. It is not intended for typical APKs, but we may find ***.jar*** files (framework code) and ***.apk*** framework overlays or UI packages (e.g., framework-res.apk, core-libart.jar).

As presented in Table 3.3, apps and libraries in the **/system** and **/system_ext** partitions can use public, system, and hidden APIs. As for apps and libraries in **/product** and **/vendor** partitions, they are allowed to use only public and system APIs.

Table 3.3: Accessibility of Android APIs depending on partition location [2]

API	**/system**	**/system_ext**	**/product**	**/vendor**	**/data**
Public API	✓	✓	✓	✓	✓
@SystemApi	✓	✓	✓	✓	✗
@hide API	✓	✓	✗	✗	✗

Many apps for some brands have been pre-installed based on the regional needs. For example, the supply chain of Transsion devices suggests that Transsion works with partners in Africa, which could pre-install apps and services tailored for the regional needs [102]. Since many steps exist to pre-install apps, this could be an entry point for introducing malicious apps on these devices [103].

3.5 Methodology

3.5.1 Research questions

Understanding the potential risks of pre-installed apps is essential. To guide our research, we have established the following key questions.

RQ1: To what extent do pre-installed apps leak sensitive data on low-cost devices? This question focuses on checking whether pre-installed apps perform activities, such as leaking sensitive data via the Internet or using other ways. This helps us to understand not just if these apps pose risks, but also how they do so.

RQ2: To what extent do pre-installed apps exhibit suspicious behaviors on low-cost devices? The aim of this question is to determine how widespread suspicious pre-installed apps are on affordable devices. Although previous research has shown that some of these apps may contain harmful code [83], a more comprehensive assessment is needed to quantify the extent of the problem in different manufacturers and regions.Through this question, we will explore the devices to identify apps having suspicious behaviors (including malware) as well as those using malicious URLs.

RQ3: How prevalent are security misconfigurations in the manifest files

of pre-installed apps on low-cost devices? Through this question, we explore apps to identify those exposing sensitive data across the exported components.

By addressing these research questions, our study provides a clearer picture of the security landscape around pre-installed apps on low-cost mobile devices and highlights potential areas where better security measures are needed.

3.5.2 PiPLAnD Design

This section outlines the workflow of *PiPLAnD*, a pipeline designed to inspect pre-installed apps using static analysis approaches. Figure 3.3 gives an overview of the pipeline. The source code of *PiPLAnD* is available on our GitHub repository[2]. We present the details of the workflow of each module in the following sections.

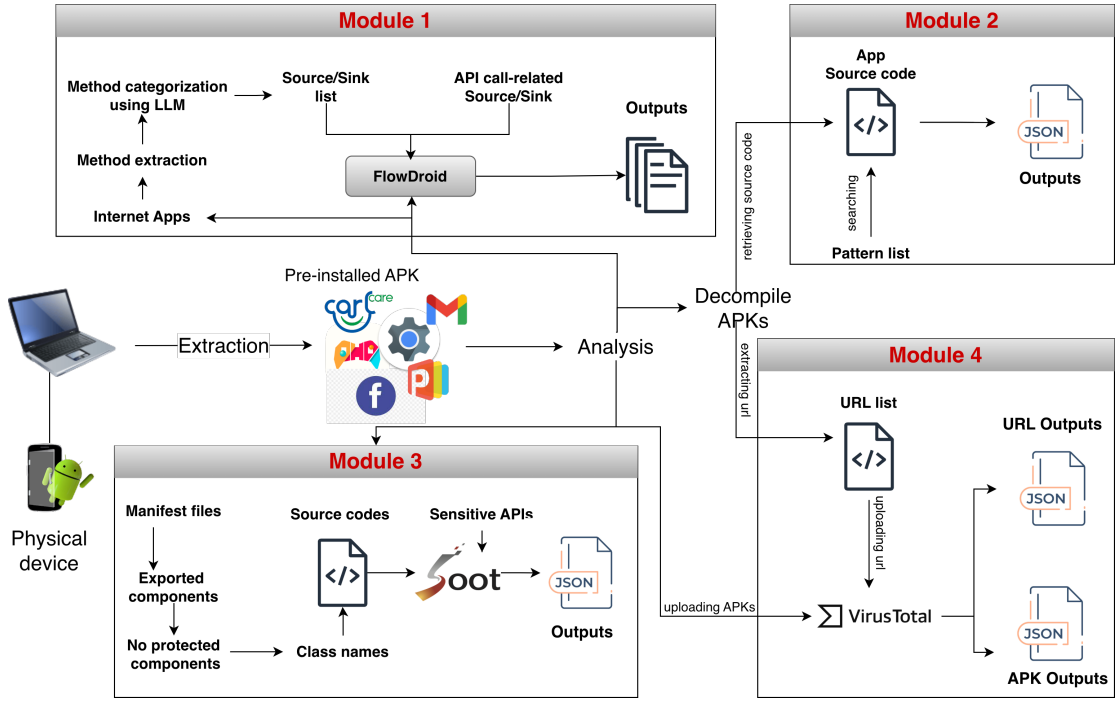


Figure 3.3: Overview of *PiPLAnD*'s workflow.

Data collection. We have used nine (9) low-cost Android devices in this study. *PiPLAnD* first automatically extracts the pre-installed apps from a physical device using ADB commands when we plug the device into the computer. After extraction, we got a dataset of 2050 APK files from the overall devices. This dataset includes system and third-party apps.

Module 1: Data leak detection.
We integrate FlowDroid into *PiPLAnD* to identify pre-installed apps that leak sensitive data. FlowDroid is based on a list of sources and sinks for the detection. This list contains Android API calls and Java methods. Thus, by default, it cannot detect data leaked from methods other than Android API calls and Java methods. During our study, we have noticed that there are pre-installed apps that use custom methods from third-party libraries to send data over the Internet. Because of that, we have identified apps that access to Internet by looking for *android.permission.INTERNET* permission on their manifest files. For these apps, we extracted all their methods

[2]PiPLAnD source code: `https://github.com/liounea/PiPLAnD`

that we have given to LLMs for source and sink categorization. The resulting list is used, in combination with the default list of sources and sinks, to identify Internet apps leaking sensitive data over the Internet. The default Source and Sink list is also used for detecting other types of data leakage in other apps.

Module 2: Behavior analysis.
The behavior analysis focuses on pattern detection. More specifically, we looked for patterns such as *"pm install"*, *"installPackage"*, etc., to detect installation behaviors, *"logcat"* for apps collecting log data, *"content://sms", "delete", "sendTextMessage", "Telephony.SMS_RECEIVED"* for accessing SMS, deleting, sending, and listening to received SMS, and so one. Table 3.4 shows the full list of the patterns we used. *PiPLAnD* helps with this analysis by decompiling the APK file using the Androguard

Table 3.4: The list of patterns

Behaviors	**Patterns**
Access / Delete / Send SMS	content://sms delete sendTextMessage Telephony.SMS_RECEIVED Telephony.Sms
Dangerous commands	"am start " "chmod " "su " "sudo " "rm -rf "
Log collection	logcat
Cliboard content access	getPrimaryClip
Installation indicators	"pm install " installPackage

framework and by checking for patterns that match our list in the app code. We only consider these patterns since they are the most obvious and are based on the existing literature, as well. Then, it reports the findings in a JSON file. This file is used for a manual analysis to confirm the behavior of the app based on the patterns detected in its code.

Module 3: Security misconfigurations on Manifest files.
This module allows for the identification of security misconfigurations in the Android manifest files. In this study, we only focus on identifying components exported that allow access to sensitive information. Android allows restricting access to an exported component by using permissions. We first extracted exported components from the manifest files for each app. Then, we filter out and keep those who did not have permission to protect and restrict their access. This gives us a list of class names with their full path in the app code. Besides that, we have constructed a list of sensitive API calls based on the source and sink list from FlowDroid, and from the Androwarn study [104], in which authors look for sensitive data accessed. We explored the list containing the classes of the exported components, and for each class, we analysed it using the SOOT framework [105]. For the analysis with SOOT, we used the sensitive API list and looked for each of them in the code of the corresponding class to know whether this component allows access to sensitive

data. If we did not find the presence of a sensitive API in the code, our analysis program constructs a call graph (CG) of the corresponding class to search whether the sensitive API is used in a method called in the code of the exported component. The results are put into a JSON file, in which we have the corresponding class, the sensitive API, and the method where we found this API.

Module 4: Maliciousness of the apps.
In this module, the goal is to point out the scope of the maliciousness of the extracted applications. To do so, we first analyze APK files extracted to identify potential pre-installed malware by uploading them to VirusTotal. We'll consider malicious only apps detected by at least 2 antivirus. Additionally, we do the same with URLs extracted from the APK files. This module enables the extraction of URLs from APKs and their analysis on VirusTotal to identify malicious URLs used by pre-installed apps. Every single URL is uploaded to VirusTotal to have the score (number of antivirus that detects it) of its maliciousness.

3.6 Analysis Results

In this section, we present the findings from nine (9) identified affordable devices that are primarily used in Africa. The main goal of this work is to evaluate the security of the pre-installed apps and to investigate the prevalence of suspicious ones on low-cost devices. Our analysis has revealed interesting findings. Table 3.5 summarizes these findings. This section presents the results of the analysis, addressing the various research questions.

Table 3.5: Summary of the results.

Analysis Modules	**Behaviors**	**# of apps (%)**
Exported sensitive components	Components that allow to access sensitive data	380 (19%)
Leak of sensitive data	Leak of sensitive data	358 (17%)
Suspicious behaviors	Dangerous commands	310 (15%)
	Malicious URLs	24 (1.2%)
	Log collection	14 (0.7%)
	Silent installation behaviors	53 (2.6%)
	Read data from clipboard	512 (29%)
	Access / Send / Delete SMS	121 (6%)

3.6.1 RQ1: To what extent do pre-installed apps leak sensitive data on low-cost devices?

Mobile apps often handle sensitive data on the device. With the privileges that pre-installed apps have, they can perform harmful activities, including leaking this data. Our analysis, consisting of identifying apps that leak sensitive data, has identified several pre-installed apps on 9 different devices leaking data, such as Mobile Country Code (MCC), user location (longitude and latitude), device info, International Mobile Subscriber Identity (IMSI), IMEI (International Mobile Equipment Identity), etc. Sensitive data is leaked in different ways, such as in SharedPreferences, in logs, in Intents, as well as on the network. Overall, we have found around 17% of the pre-installed apps on the devices leaking sensitive data, which represent 358 pre-installed apps. This number is shared among the 9 devices.

Indeed, as illustrated in Fig. 3.4, each low-cost device contains a least 4% of the pre-installed apps leaking data. As an example, the app (*com.transsion.statisticalsales*)

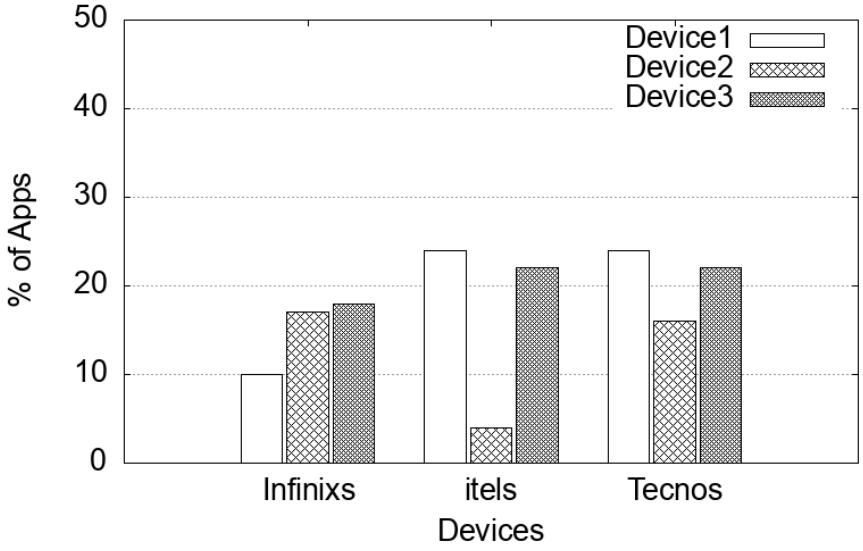


Figure 3.4: Percentage of apps leaking data on the devices

sends sensitive information to a remote host by using third-party libraries with customized methods. As illustrated in Listing 1, the app collects location info, IMSI, IMEI, phone version, etc., and sends them to a remote server. Pre-installed apps also leak data through other means, such as SharedPreferences storage and device logs. These practices expose users to numerous security and privacy issues.

Answer to RQ1

The results show that several pre-installed apps on the three devices exhibit harmful activities such as leaking sensitive data, exposing the users to severe risks.

3.6.2 RQ2: To what extent do pre-installed apps exhibit suspicious behaviors on low-cost devices?

Since mobile devices come with pre-installed apps, these may contain harmful code that can compromise the security of users [83]. We have looked for apps having suspicious behaviors. Several pre-installed malware have been identified to have silent installation behaviors [83]. Others steal sensitive data using different ways [83, 106, 107]. To identify these kinds of malware, we focus the analysis in detecting patterns.

When a pre-installed app declares the *INSTALL_PACKAGES* permission, it has the ability to install an app without the user's knowledge. It can use *android.content.pm.PackageManager.installPackage()* or *Runtime.exec('pm install')* to silently install the app [83]. Our analysis has revealed 53 pre-installed apps having this silent installation behavior on the overall devices. Several pre-installed apps have access to the SMS provider *content://sms*. Some of them delete SMS

```
[...]
public class SSHttpClient {
    public static final String BASE_URLFLAG = "PCHttpClient";
    private static final String DEFAULT_BASE_SERVER =
    ↪ "https://asv.transsion.com:443/SaleStatistics/sendsale/sendSale";
    private static final String DEFAULT_INDIA_SERVER =
    ↪ "https://asvin.transsion.com:8080/SaleStatistics/sendsale/sendSale";
    protected static final String TAG = "SSHttpClient_";
    [...]
    public void RegisterInformation(final HttpCallback<HttprequestResult> httpCallback, boolean
    ↪ z) {
        RequestParams requestParams = new RequestParams();
        requestParams.put("ua", this.requestInfo.getUa());
        requestParams.put("screen", this.requestInfo.getScreen());
        requestParams.put("imsi", this.requestInfo.getImsi());
        requestParams.put("imei", this.requestInfo.getImei());
        requestParams.put("phone_version", this.requestInfo.getPhone_version());
        requestParams.put("platform", this.requestInfo.getPlatform());
        requestParams.put("device", this.requestInfo.getDevice());
        requestParams.put("lang", this.requestInfo.getLang());
        requestParams.put("timeStamp", this.requestInfo.getTimeStamp());
        requestParams.put("auth", this.requestInfo.getAuth());
        requestParams.put("lat", this.requestInfo.getLat());
        requestParams.put("lng", this.requestInfo.getLng());
        requestParams.put("client_type", this.requestInfo.getClient_type());
        requestParams.put("phone", this.requestInfo.getPhone());
        requestParams.put("client_version", this.requestInfo.getClient_version());
        requestParams.put("lac", this.requestInfo.getCELL_LAC());
        requestParams.put("cid", this.requestInfo.getCELL_CID());
        mClient.post(z ? DEFAULT_INDIA_SERVER : DEFAULT_BASE_SERVER, requestParams, new
        ↪ AsyncHttpResponseHandler() {
            [...]
        });
    }
}
```

Listing 1: App sending sensitive data to a remote server

or declare the *SEND_SMS* and *RECEIVE_SMS* permissions with the broadcast action *Telephony.SMS_RECEIVED*, allowing them to listen and send messages. We have found around 121 apps pre-installed on these low-cost devices, sending, deleting, and/or reading SMS contents. Furthermore, at least 14 pre-installed apps can access and collect the logcat content. Since they have déclared the *READ_LOGS* permission, they can access the overall logcat content, including logs from other applications. Table 3.6 presents the repartition (in %) of apps that present these behaviors over the 9 devices. It shows that each of the devices contains apps that

Table 3.6: Statistics of apps with suspicious behaviors across the devices

Devices		**Log collection (%)**	**Silent installation behavior (%)**	**Access / send / delete SMS (%)**
Infinix	Device1	0.8	2.5	7
	Device2	0.8	3	5
	Device3	0.6	2.3	8
Itels	Device1	0.8	2	6
	Device2	0.8	2.4	6
	Device3	0.4	2	5
Tecnos	Device1	0.4	2	5
	Device2	0.7	4	5
	Device3	0.8	3	6

have these behaviors. Additionally, we have identified several apps that execute dangerous commands (310 apps) and others that can access and read the clipboard content (512 apps). Fig. 3.5 shows the graph presenting the percentage of apps that execute dangerous commands on each device. We notice that each device has at least 5% of the apps executing dangerous commands. Similarly, Fig. 3.6 presents the

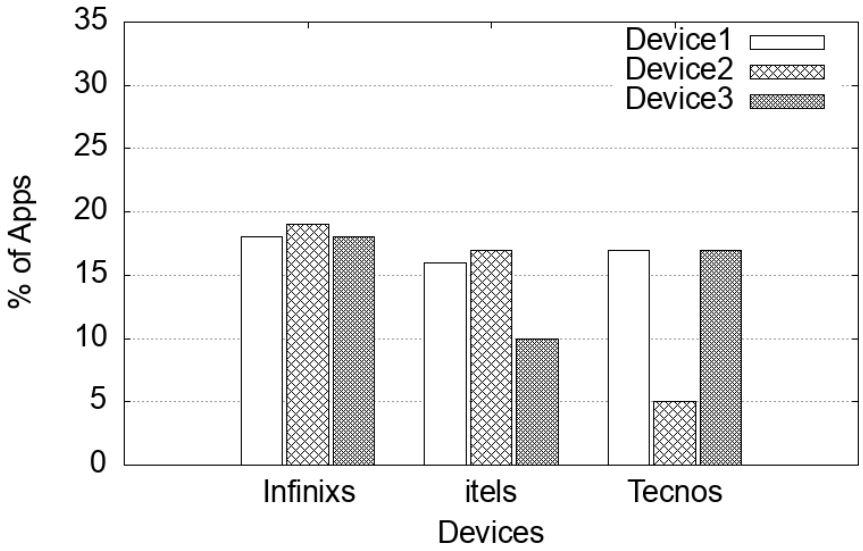


Figure 3.5: Percentage of apps executing dangerous commands

percentage of apps accessing the clipboard data. It shows that between 8 and 33% of the apps in low-cost devices access the content clipboard.

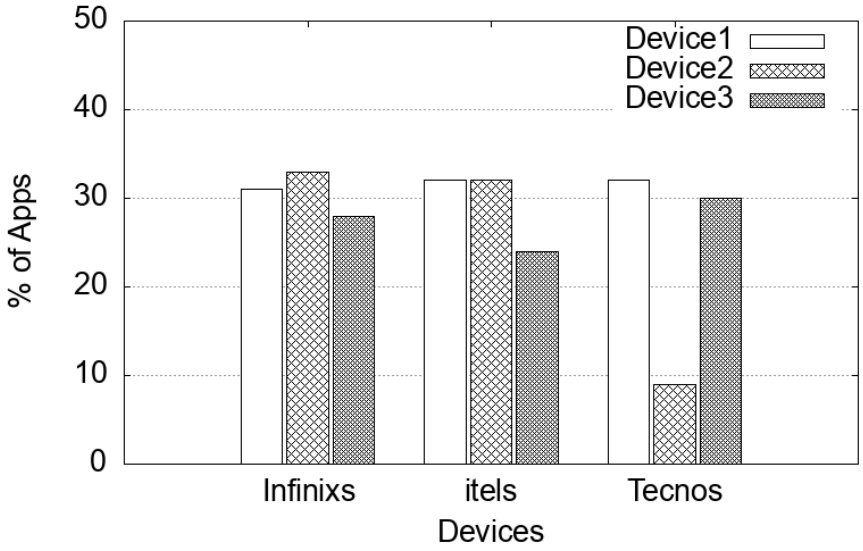


Figure 3.6: Percentage of apps accessing clipboard content

Furthermore, we have extracted the URLs used by the apps from the APK files for each device. Then, we have analysed them on VirusTotal to identify the potentially malicious ones. We considered a URL as malicious only if it is detected by at least two (2) VirusTotal antivirus (Score $\mathbf{s} \geq \mathbf{2}$). Table 3.7 illustrates the number of malicious URLs in each brand and their scores (the number of antivirus that tagged

Table 3.7: Malicious URLs detected per brands and VirusTotal score ranges

Device brands	**# of malicious URL**	**# of URL w/ score (s)**		
		$2 \leq s \leq 4$	**$5 \leq s \leq 7$**	**$s \geq 8$**
Infinixs	19	13	3	3
itels	9	7	0	2
Tecnos	6	4	2	0

the URLs as malicious). As a result, we have identified 24 apps that use at least one malicious URL on overall devices, as shown in Table 3.5. As illustrated in Table 3.7, there are several malicious URLs used by apps from Infinix (19), itel (9), and Tecno (6) devices. Most of these URLs have a score between 2 to 4. However, there are a small number of URLs with a score of at least 8. Similar to the URL analysis, we have also analysed the APK files on VirusTotal to detect potential malware. The results showed that 10 APK files (4 from Infinix, 3 from itel, and 3 from Tecno devices) have been detected as malicious by one antivirus each. However, since we only considered as malicious apps those that have been detected by at least two antivirus programs, we did not consider these apps as malware.

Answer to RQ2

Low-cost Android devices ship pre-installed apps with suspicious behaviors, including sending/deleting/reading SMS, executing dangerous commands, reading clipboard content, accessing overall logcat memory, and having silent installation behaviors. Furthermore, several pre-installed apps also use malicious URLs.

3.6.3 RQ3: How prevalent are security misconfigurations in the manifest files of pre-installed apps on low-cost devices?

Exported sensitive components. Android apps often export components, such as activities, services, receivers, and providers, explicitly by setting *android:exported="true"* or implicitly by declaring an *intent-filter* in the manifest file. When it is the case, the app allows other apps to launch the component [93]. The access to exported components is often restricted using permissions [108]. If there is permission for an exported component, the app that wants to launch it should declare this permission. If an app exports a component without properly enforcing permission, any app could launch it or access sensitive data it contains [109]. When a component allows access to sensitive data, we call it a sensitive component. Our analysis has revealed that several pre-installed apps have sensitive components exported. As presented in Table 3.5, we have found around 19% of the pre-installed apps, representing 380 different app versions, having exported sensitive components on the low-cost devices, without any restriction or protection mechanism. Fig. 3.7 shows that every device exports sensitive components, with an Infinix device leading the trend. It means that these

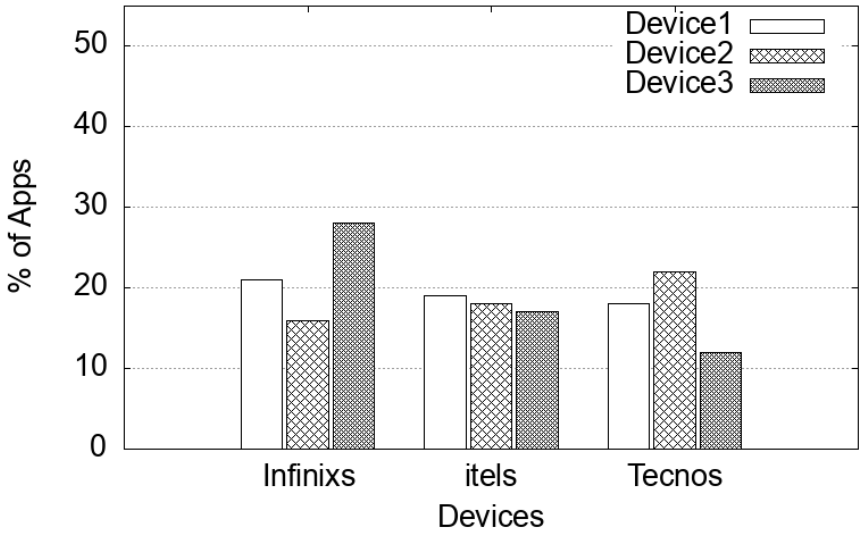


Figure 3.7: Percentage of apps exporting sensitive components without any protection

devices embed apps that allow other apps to potentially access sensitive information, exposing the user to security and privacy problems. For example, we have found a pre-installed app (*com.transsion.carlcare*) having this method (Listing 2), from an exported activity (*com.transsion.carlcare.WarrantyCardActivity*). This method allows access to location information (lines 13 and 14). In the Android manifest file of the app, this component is clearly exported; however, it is protected by no mechanism, facilitating its easy access and its potential exploitation. Another example shows an exported ContentProvider (*com.sprd.providers.photos.SpecialTypesProvider*) found in the app (*com.sprd.providers.photos*) that allows access to media files, contained in an external storage, from a URI (Listing 3, line 10). This exported component does not have a mechanism that restricts its access by other apps.

```
1  public void R1() {
2      [...]
3      HashMap<String, String> map = new HashMap<>();
4      if (TextUtils.isEmpty(this.f15263g0)) {
5          map.put("imei", listA.get(0));
6      } else {
7          map.put("imei", this.f15263g0);
8      }
9      map.put("imsi", wd.c.g());
10     map.put("lang", getResources().getConfiguration().locale.
11                                         toString());
12     if (this.f15282z0 != null) {
13         map.put("lat", this.f15282z0.getLatitude() + "");
14         map.put("lng", this.f15282z0.getLongitude() + "");
15     }
16     map.put("phone_version", a2());
17     map.put("screen", getResources().getDisplayMetrics().widthPixels + "*" +
18                             getResources().getDisplayMetrics().heightPixels);
19     map.put("ua", Build.BRAND + "-" + Build.MODEL);
20     [...]
21 }
```

Listing 2: Exported activity accessing sensitive information

```
[...]
public final class SpecialTypesProvider extends ContentProvider {
    [...]
    private static final Uri EXTERNAL_CONTENT_URI = MediaStore.Files.getContentUri("external");
    private static final String[] SPECIAL_TYPE_PROJECTION = {"_data", "owner_package_name"};
    [...]
    private int getCameraType(long j) {
        Log.d(TAG, "mediaStoreId = " + j);
        boolean z = true;
        Cursor cursorQuery = getContext().getContentResolver().query( EXTERNAL_CONTENT_URI, SPECIAL_TYPE_PROJECTION,
                             "_id=?", new String[]{String.valueOf(j)}, null);
        if (cursorQuery != null) {
            try {
                if (cursorQuery.moveToFirst()) {
                    String string = cursorQuery.getString(0);
                    Log.d(TAG, "mediaPath = " + string);
                    String string2 = cursorQuery.getString(1);
                    if (!GOOGLE_PHOTOS_PACKAGE_NAME.equals( string2) && !GOOGLE_GALLERY_PACKAGE_NAME.equals( string2)) {
                        z = false;
                    }
                    Log.d(TAG, "ownerPackageName = " + string2 + ", isGoogleCreatedImage = " + z);
                    try {
                        ExifInterface exifInterface = new ExifInterface();
                        exifInterface.readExif(string);
                        iIntValue = z ? 0 : exifInterface.getTagIntValue( ExifInterface.TAG_CAMERATYPE_IFD). intValue();
                        Log.d(TAG, "getCameraType cameraType = " + iIntValue);
                    } catch (Exception e) {
                        Log.d(TAG, "Exception occurs, mediaPath = " + string + ". ex = " + e);
                    }
                }
            } finally {
                if (cursorQuery != null) {
                    cursorQuery.close();
                }
            }
        }
        return iIntValue;
    }
    [...]
}
```

Listing 3: Exported ContentProvider accessing external storage files

Answer to RQ3

The results have revealed several pre-installed apps exporting sensitive components, including activities, services, content providers, and receivers, on low-cost devices. This potentially puts users at risk, as their data could be accessed by third parties.

3.7 Discussion

3.7.1 Discussion about findings

In this section, we discuss the results from the analysis of the devices.

Sensitive data leakage. During our analysis, we have found several pre-installed apps leaking sensitive data. Several of them send the data to a remote host by using Android API calls. Others use third-party libraries that use customized methods to avoid existing detection (e.g, *com.transsion.statisticalsales*). We have considered these customized methods and added them to FlowDroid, and we have detected apps sending sensitive data over the Internet with these methods. Considered as malware by some blog posts [79], the app (*com.transsion. statisticalsales*) is not detectable by malware scanners such as VirusTotal. It silently collects data (phone version, IMSI, user location, CID (Cell Tower ID), LAC (Location Area Code), etc.) and sends it to a remote host. This suspicious behavior compromises users' security and privacy. Several other apps get and store sensitive data in internal storage, such as SharedPreferences, or log it in the logcat memory. These leaked data can be used by malicious actors for user profiling, user tracking [110], or accessed by other apps since

pre-installed apps can share each other's data when they have the same *sharedUserId* and have signed with the same certificate [83].

Suspicious behaviors. We have found several pre-installed apps having suspicious behaviors, including silent installation behavior, send/delete SMS, etc. These are done without the user's knowledge and may have negative consequences. Indeed, when an app is able to install another app silently, it may install a malicious app from a malicious remote server or via dynamic code loading. This technique is often used by malicious actors to infect Android devices and avoid detection [83]. This is possible only with system apps, since non-system apps cannot declare the required permission. When a pre-installed app has the ability to send and delete messages without the user's interaction, this can lead to the theft of sensitive data. Several malware are known to use this technique. For instance, a variant of the Triada malware has been found pre-installed in Android devices [106]. This malware performs several actions, including enabling premium SMS services, intercepting, sending, and deleting messages. Additionally, our analysis of the pre-installed apps has revealed that several of them (around 24 apps) use malicious URLs. Some of those URLs have been detected by more than five (5) antivirus programs, indicating the level of compromise of these apps. In summary, several applications could compromise the user's security without their knowledge, as they exhibit suspicious behaviors.

Security misconfiguration. Our analysis has revealed several apps exporting components that allow other apps to potentially access sensitive data, without any protection. The Android framework proposes permission levels, including normal, dangerous, and signature, to protect and restrict access to components [111]. The absence of protecting exported components is a known vulnerability from the community [111]. When an app exports a component without any restriction, it allows other apps to access it. From this, a malicious app may access sensitive resources, as mentioned in the Common Weakness Enumeration (CWE - 926). The app can be victim to an attack named confused deputy attack [112, 88], as well as a malicious app can abuse the privileges that these components have to gain unauthorized access to these resources [111, 113].

3.7.2 Use cases

During our analysis, we have identified pre-installed applications having particular behaviors. In this section, we go deeper into them by explaining their behaviors.

3.7.2.1 SalesStatistics (com.transsion.statisticalsales)

Our findings about *SalesStatistics* app and considering the fact that it has been mentioned in a blog post [79] have encouraged us to perform further manual analysis to have a deep look into it. The behavior of the app allows us to consider it as malicious (malware). This malware is not detectable by VirusTotal's antivirus. The application named `SalesStatistics` with the package name `com.transsion.statisticalsales` has been found pre-installed on the three devices we focused on. It has been located and extracted from the **/system_ext/app** folder on the devices. `SalesStatistics` has the same sharedUserId (*android.uid.system*, which allows apps to have the system-level privileges) and the same signature certificate as at least 15% of the pre-installed apps on each device. This means that it can access and share data with at least 15% pre-installed apps [83]. This app can access to Internet (with the *android.permission.INTERNET* permission) on the Infinix device to send/receive data, but it cannot do the same on itel and Tecno. The

user cannot interact directly with this app since it does not have a user interface.

In the Medium blog post[79], Bobe mentioned that this malware (`SalesStatistics`) sends sensitive information to a remote server from Infinix, itel, and Tecno devices. Indeed, through a service called *RegisterService*, it collects information, including LAC, CID, IMSI, IMEI, location info, etc., and sends it to remote servers (**"https://asv.transsion.com:443"** and **"https://asvin.transsion.com:8080"**). However, this behavior is only found in the Infinix device, while in itel and Tecno, this app does not have the INTERNET permission; thus, it cannot send data over the Internet.

In addition, we have found the app accessing the SMS content, sending and deleting messages on the three devices. It has declared the *"android.permission.SEND_SMS"* permission and uses *android.telephony.SmsManager.sendTextMessage()* to send SMS messages containing sensitive information. ***sendTextMessage(getSmsNum(), null, sMSString, sendPI2, deliveryPI2)*** is an instruction that it uses, among others, to send data. In this example, the variable ***sMSString*** contains data, such as IMEI, IMSI, phone number, phone version, GSM location area code (LAC), GSM cell ID (CID), locale language, and other information about the device. After that, it deletes all the messages related to the destination phone number from *"content://sms"* using the *ContentResolver.delete()* method.

3.7.2.2 TPMS (com.hoffnung)

We have found the app named *TPMS (pakacge: com.hoffnung)* having suspicious behaviors that draw our attention. We have performed a manual analysis to confirm these findings, and we have found that this app runs persistently in the background. It is installed on the three devices and has been placed into the **/system_ext/app** folder. The app tracks the app currently in the foreground using *onForegroundActivitiesChanged()*. If a target app is opened, it sends an external event (*sendExternalEvent*). For example, the app monitors the clipboard to detect changes via the *onClipboardChange()* method. To do so, it checks if it is running on an API level version lower than 30. If it is the case, it accesses and logs the clipboard content *(Log.i("AthenaDelegator", "trackClipboard pkgName: " + str + " text: " + str2))*, and sends it via a broadcast receiver (*sendBroadcast()*) if certain conditions are met. This means that other receivers inside the same app (or a privileged companion app) will receive this data and process it (possibly to forward it to a remote server). This data transmitted may include sensitive user data, allowing this app to play a key role in transmitting it from the device to analytics or surveillance services embedded by the device vendor. In addition to this, the app monitors changes in audio settings and user accounts, and logs them after a short delay. It can programmatically force apps to stop (*forceStopPackage()*).

3.8 Summary

This work investigates and analyzes Android pre-installed apps, in particular, those shipped with phones sold in Africa. For this purpose, we have proposed a pipeline that allows the extraction of APK files from a physical device and inspects them to look for different suspicious behaviors, including pre-installed malware, apps exposing sensitive data, and apps sending personal data to remote hosts. The pipeline is tested by inspecting nine different low-cost Android devices bought in

Africa. The results show interesting findings, such as ① the leak of sensitive data, ② sensitive data exposure through exported components, and ③ apps having suspicious behaviors.

As future research, we planned to go deeper into these pre-installed apps by analyzing the native libraries used and by looking for further suspicious behaviors. Our future study will extend the number of low-cost devices and compare the results from the analysis of pre-installed apps with those from European devices, as well. In addition, we planned to perform a dynamic analysis by implementing a solution that monitors the system log and detects suspicious behaviors using AI models.

4

Security Assessment of Mobile Banking Apps in West African Economic and Monetary Union

In this chapter, we evaluated fifty-nine (59) West African Economic and Monetary Union (WAEMU) mobile banking apps (MBAs) using static analysis techniques. We collected these MBAs from the 160 banks and financial institutions of the eight WAEMU countries listed on the website of the Central Bank of West African States (BCEAO). We identified security-related code issues that malicious actors could exploit. We investigated the issues found in the older versions to track their evolution across updates. Additionally, we identified some banks from regions such as Europe, the United States, and other developing countries and analyzed their MBAs for a security comparison with MBAs from WAEMU. Key findings include: (1) WAEMU apps exhibit security issues introduced during development, posing significant exploitation risks; (2) Despite frequent updates, underlying security issues often persist; (3) Compared to banking apps from developed countries, WAEMU apps exhibit fewer critical issues; and (4) Apps from banks that are branches of other non-WAEMU banks often inherit concerns from their parent apps while also introducing additional issues unique to their context. Our research highlights the importance of robust security practices in WAEMU MBA development to promote user safety and trust in financial services.

This chapter is based on the work published in the following research paper [114]:

- A. Diallo, A. War, M. A. Diouf, J. Samhi, S. Arzt, T. F. Bissyandé, and J. Klein, "Security Assessment of Mobile Banking Apps in West African Economic and Monetary Union," 2025 Cybersecurity4D (C4D), Gqeberha, South Africa, 2025, pp. 1-13. https://doi.org/10.1109/C4D65382.2025.11306468.

Contents

4.1 Overview

The financial sector plays a pivotal role in the economic development of any country, with digital financial services, including mobile money and mobile banking, emerging as powerful catalysts for this progress, particularly in Sub-Saharan African nations. The widespread adoption of mobile banking is a global phenomenon, with Sub-Saharan Africa leading the way in its innovative implementation [12]. This surge can be attributed mainly to the remarkable proliferation of smartphones, a trend projected to continue its upward trajectory. Nowadays, approximately 60% of people have smartphones worldwide [115]. By 2022, smartphone adoption in Sub-Saharan Africa had reached an impressive 51%, with projections indicating a substantial rise to 87% by 2030 [116].

Recognizing the strategic advantage of widespread smartphone penetration, banks and financial institutions within the West African Economic and Monetary Union (WAEMU) region are leveraging this technology to extend banking services. This leap serves a dual purpose: broadening access to banking services for millions and empowering existing account holders with remote control over their finances, facilitating transactions and balance checks. In WAEMU, nearly all banks and financial institutions have embraced mobile applications, reaching millions of people reliant on these platforms for daily financial transactions.

However, the ubiquity of mobile banking applications is accompanied by recurring security concerns, posing challenges for users and institutions alike. Reports highlight, for example, the persistence of sensitive keys hard-coded into financial mobile applications across Africa [117], underscoring the potential for significant economic losses. Despite the growing reliance on these applications in developing countries, including in the WAEMU, comprehensive studies on their security remain scarce. A literature review reveals limited studies on mobile financial applications in developing nations, with no research specific to the WAEMU region [18].

Various approaches have been explored to assess the security issues of mobile banking apps, ranging from investigating their origins [62] to conducting forensic examinations and vulnerability assessments [63, 35]. However, there remains a gap in research addressing these concerns comprehensively.

Against this backdrop, our work extensively investigates the security landscape of mobile banking applications in WAEMU countries. Our research aims to identify prevalent security issues, establish a comprehensive threat model, analyze the evolution of security across different app versions, and compare the security posture of WAEMU banking apps with those from the European Union (EU), the United States (US), and other developing countries (ODC). Moreover, some WAEMU financial institutions are branches (children) of other institutions (parents). We explore the security of their mobile banking apps and determine if they inherit parent app security issues.

The contributions of this study are as follows:

- Highlighting the most common security problems affecting WAEMU banking apps.
- Providing possible ways to exploit app vulnerabilities to understand the risks better.
- Offering an assessment of the security evolution in these apps, including whether concerns are effectively addressed in subsequent updates.

- Offering a comparative analysis of the security posture of WAEMU banking apps in the global context, comparing them with the applications used by leading African and International banks.
- Shedding light on the security problems between the branch and parent apps.

These findings underscore the imperative of enhancing mobile banking app security to safeguard users and institutions in an increasingly digital financial landscape.

Artifacts. We release all of our artifacts (e.g., list of mobile apps, detailed results, etc.):

`https://github.com/liounea/Data_for_WAEMU_Apps_Papers`

4.2 Background and Related Works

4.2.1 Background

4.2.1.1 What is mobile banking?

In this work, we excluded mobile money apps that are not necessarily related to a bank account. We indeed focus on mobile banking, which offers services that allow customers to use their bank account through mobile equipment [118]. These services could be bank transfers from one account to another, balance checking, or bill payments. In general, banks develop applications that could be used on Smartphones to facilitate the use of their services. Those applications are called mobile banking applications or mobile banking apps (shortly MBAs here).

4.2.1.2 What are security code smells and vulnerabilities?

Security code smells (code smells or smell) are security issues from a bad implementation and poor design in software [119, 120]. They are not necessarily exploitable. However, they could have consequences and could compromise the security and privacy of software users [121]. In this case, they are called vulnerabilities.

4.2.2 Related Works

To our knowledge, no study has focused on mobile banking apps in WAEMU countries. However, several works have been performed on mobile banking and money apps in other contexts.

Bassolé et al. [34] investigate Android banking and payment apps in African countries by identifying specific vulnerabilities and raising awareness of the security in those apps. The authors provide valuable information to developers and stakeholders, enabling them to improve security measures and protect user data.

Bowers et al. [122] perform a security analysis of digital credit apps in developing countries. In their study, they investigated the privacy policies of several companies and found that previously undisclosed data types were collected. They investigated the configurations between apps and servers, as well, and they discovered widespread misconfiguration of encryption.

In their work, Latifa et al. [66] investigate the impacts that permissions could have on the security of users' mobile banking apps. By analyzing the permissions requested by some mobile banking apps from Maghreb countries, they found that many apps use unnecessary and dangerous permissions.

Bojjagani and Sastry [123] propose a threat model (VAPTAi) for enhancing the security of Android and iOS mobile banking apps. VAPTAi is designed to assess

vulnerabilities and penetration testing of mobile banking apps. This work is a bit similar to one of their prior work in which the same authors propose a tool that identifies threats at different levels, including app, network, and device levels [124]. In this work, they analyze vulnerabilities using static, dynamic, and forensic analysis techniques and identify several attack surfaces.

Kaka et al. [68] evaluate the vulnerability of mobile banking apps from India by mainly performing man-in-the-middle (MiTM) attacks. In their work, the authors discover that in most apps, MiTM attacks could be successfully achieved even if the apps use HTTPS for communication with the servers.

Castle et al. [125] evaluate the security challenges of mobile money systems in the developing world by identifying vulnerabilities, assessing the factors contributing to them, and proposing potential solutions to enhance the security of such systems. In the same paper, the authors have given a response to a prior work in which Reaves et al. [126] identify and document the security and privacy issues in branchless banking apps to raise awareness among developers, financial institutions, and regulators about the importance of robust security measures.

4.3 Methodology

This work assesses the security of MBAs from financial institutions in the WAEMU countries. This section describes the Research Questions (RQs) and the app collection and analysis process.

4.3.1 Research questions

Our study aimed to provide a comprehensive understanding of the security landscape surrounding mobile banking apps in the WAEMU region. To achieve this, we formulated 4 RQs.

RQ1: To what extent do mobile banking apps from WAEMU present critical security issues?

We have scanned previously collected banking apps for vulnerabilities to address this RQ. We have found several critical issues in WAEMU banking apps. We have confirmed some of these critical issues as vulnerabilities by manual verification and have examined real-world attack scenarios for some of these critical vulnerabilities. This research question reveals that banking apps in the WAEMU are highly vulnerable and can be easily compromised. Hence, practitioners must recognize and address these issues to make mobile banking safer in the WAEMU.

RQ2: How do security issues evolve in WAEMU bank apps? Given the ever-evolving landscape of technology, the discovery of new security issues and vulnerabilities in mobile apps is a persistent concern. This question investigates the responsiveness of WAEMU banks to these security concerns. It investigates the practice of proposing and implementing regular updates to address these concerns and scrutinizes whether these updates effectively rectify issues in subsequent app versions.

RQ3: How vulnerable are WAEMU banking apps compared to apps from 1) the top 20 EU banks, 2) the top 20 US banks, and 3) the top 20 other developing countries' banks? Our study extends its purview to include a comparative analysis of WAEMU banking apps against those of top European Union and United States banks and top banking apps from other (i.e., non-WAEMU) developing countries. This comparative assessment serves several critical purposes:

it elucidates whether these apps share similar security issue profiles, discerns the disparities in vulnerability severity and distribution, and identifies factors that render WAEMU banking apps more or less vulnerable than their international counterparts.

RQ4: How vulnerable are child banking apps compared to parent banking apps? Some WAEMU banks are branches of international banks. We carry out this study to investigate whether the apps of those banks (child apps) inherit the problems of the respective African and international parent banks.

Through the answers to these research questions, we provide a comprehensive examination of the security status of mobile banking apps in the WAEMU region, offering actionable insights and potential avenues for enhancing the security of these applications to safeguard the interests of both users and financial institutions.

4.3.2 App selection process

4.3.2.1 Collecting WAEMU banking apps

We meticulously considered all the 160 banks and financial institutions across the eight WAEMU countries[1] listed on the Central Bank of West African States (BCEAO) website [127]. Subsequently, we gathered a comprehensive dataset comprising only fifty-nine (59) Android banking applications from Google Play. We do not have the same number of applications as institutions (160) because several institutions have the same mobile app, and others have none. The oldest available APKs have been collected from December 15 to December 16, 2022. Our decision to focus exclusively on the Android platform stems from its predominant usage in Africa. As indicated by statistics [128], the Android operating system led the market share in Africa in November 2022, with a rate of 83.87%.

To better understand the security implications of these findings, we manually analyzed the apps' source code. This manual examination facilitated the establishment of potential real-world attack vectors for exploiting the identified vulnerabilities.

4.3.2.2 Selection of old versions of WAEMU apps

We meticulously collected historical versions of these apps to investigate the evolution of security issues across different versions of WAEMU banking apps. As mentioned, our study encompasses 59 WAEMU apps, comprising the newest available APKs. These served as reference points for analyzing the evolution of security features. We relied on AndroZoo [129] to procure the historical APKs for all 59 apps. Among the 59 apps, some (approximately 20%) do not have more than two versions. AndroZoo provides extensive metadata for each APK, of which the most important for our study are the SHA256 hash, package name, and the date of addition (the date on which the corresponding APK is added to AndroZoo). We differentiate two versions of the same app if they have the same package name and different SHA256 hashes. The oldest among the two versions is determined considering the dates of addition to AndroZoo. AndroZoo regularly crawls the Google Play Store. Hence, the date of addition to AndroZoo represents the age of the app in the store.

To facilitate a comprehensive understanding of security evolution, we aimed to select five versions for each app, including the oldest APK. The versions were chosen at an interval of two months, at least between two consecutive versions. Several apps have a significant number of versions. For those apps, we considered an interval of

[1]The eight countries are: Benin, Burkina Faso, Côte D'Ivoire, Guinea-Bissau, Mali, Niger, Senegal, and Togo

at least one year to choose versions. Some apps did not have sufficient historical versions, so we did not need to exclude versions in such cases.

Consequently, our final dataset comprises apps with varying numbers of versions, ranging from 2 to 5, inclusive of the reference version. This meticulous approach enables a nuanced analysis of security trends and evolution across different iterations of WAEMU mobile banking applications.

4.3.2.3 Selection of European Union and United States apps

For the European Union (EU) and United States (US) banking apps, we have selected the top 20 banks for each of them based on their total asset value [130, 131]. Each of the chosen banks can have many mobile apps due to their presence in many countries. Thus, we identified the package names of these banks' apps from their respective headquarters cities, as the package name serves as a unique identifier for each app.

4.3.2.4 Selection of apps from other developing countries

Apps from developing countries other than WAEMU countries have been selected, too. The goal is to compare the security of mobile banking apps within similar development contexts. To respect the same process as for developed countries, we selected the top 20 banks from developing countries, excluding WAEMU members. We based the selection criteria on the 2023 bank rankings provided by Brandirectory on its website [132], which ranks banks globally according to their total assets. Brandirectory's comprehensive ranking lists the top 500 banks in 2023, from which we identified the 20 most valuable banks in developing countries, excluding those from WAEMU.

4.3.3 Automated analysis tool

To analyze our collected MBAs, we employed a robust vulnerability scanner tool called VUSC[2]. VUSC is a commercial tool built upon the Dexpler [133] and Soot framework [105], as detailed in prior research [134]. This tool translates the app's bytecode into a more analyzable format, such as Jimple code [135], facilitating static analysis to uncover potential vulnerabilities. Furthermore, VUSC utilizes FlowDroid [136] to conduct data flow analysis, identifying potential data leakage points within the app. It is essential to note that the results generated by VUSC primarily comprise Security-Related Code Smells (SRCSs), i.e., all of them are not necessarily exploitable, as elucidated in the literature [134].

4.4 Empirical Results

This section presents the results of our empirical study. As presented above, we have collected and successfully analyzed 59 MBAs from the WAEMU countries. The VUSC scanner has reported numerous potential security issues for MBAs in WAEMU countries. As these issues are not yet validated, we call them SRCSs, a term coined in previous research [134]. We grouped these SRCSs into three categories: high, medium, and low, as shown in Figure 4.1.

[2] `https://www.sit.fraunhofer.de/en/offers/projekte/vusc/`

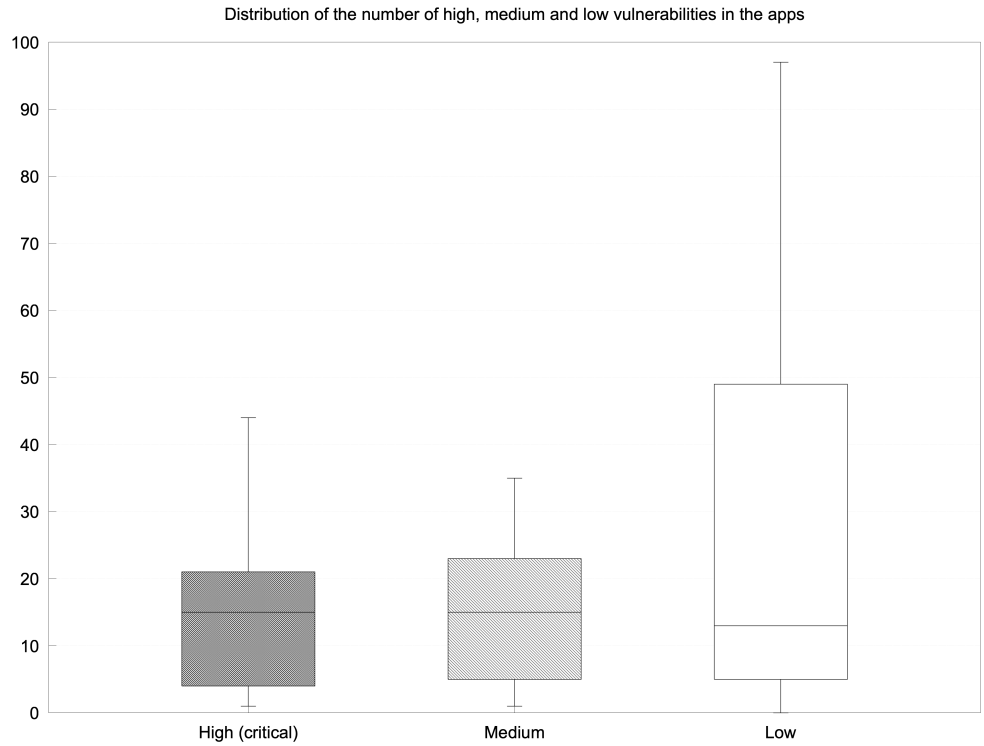


Figure 4.1: Overview of the number of vulnerabilities found in WAEMU banking apps.

4.4.1 RQ1: To what extent do mobile banking apps from WAEMU present critical security issues?

In the remainder of this work, due to the high number of findings across all apps, we focus on **the most critical ones, i.e., the ones labeled as high severity by the VUSC scanner**. The results show that each app has at least one possible security issue, and half have over fifteen SRCSs. As shown in Fig. 4.2, the ten most common SRCSs in the WAEMU banking apps affect between 20 to 80% MBAs. We next describe these highly prevalent SRCSes in detail.

Use of insecure cryptographic algorithm (80%). Cryptographic algorithms may be used for tasks such as encryption/decryption, electronic signatures and the verification thereof, computation of hash functions, and other security-critical functions. Using an insecure or outdated algorithm compromises the respective security property. In the case of encryption, attackers with enough computational power may be able to retrieve the plaintext without knowing the key via cryptanalysis or brute-force attacks.

Content provider access from WebViews (78%). Apps may display web content in a component called `WebView`. Allowing this `WebView` to interact with content providers inside the same app can be a security risk, especially if the `WebView` may be tricked into displaying untrusted content or executing untrusted JavaScript code.

Bad hostname verifier (51%). When an Android app communicates with a server over HTTPS (SSL/TLS), it typically relies on X.509 certificates to authenticate the server and establish a secure connection. During the SSL/TLS handshake, an app must verify that the server's hostname matches the hostname(s) listed in the X.509

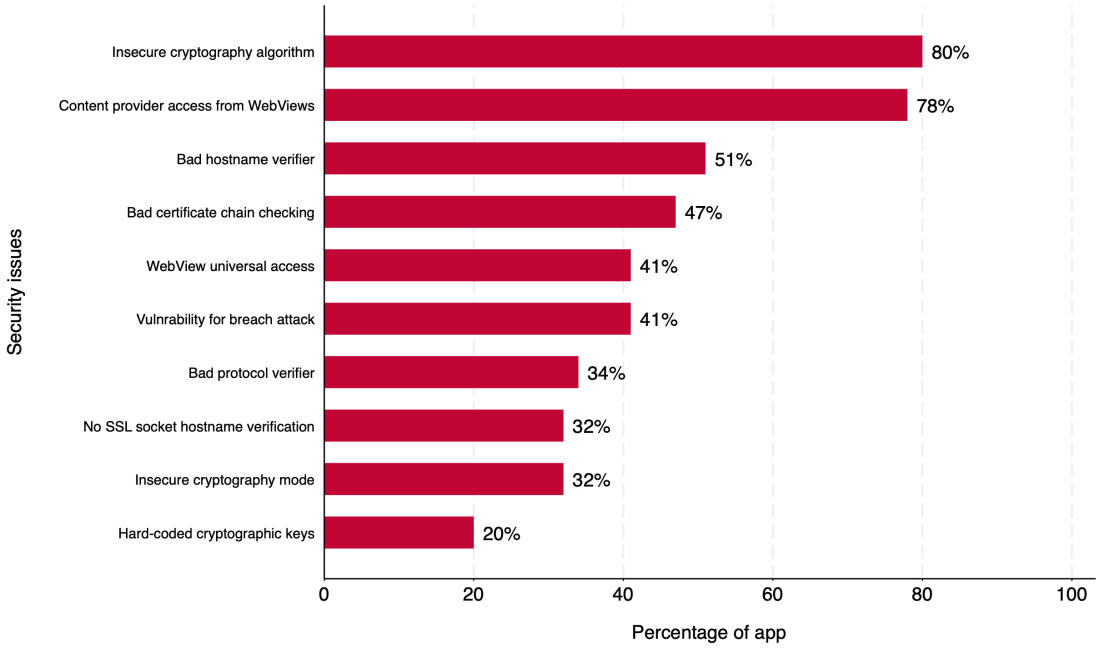


Figure 4.2: Top ten of the most common critical security issues found in the WAEMU banking apps and the percentage of apps for each vulnerability.

certificate the server presents. By default, this check is handled by the operating system. Apps may, however, implement their own verifier, e.g., to accept debug certificates during development or to cater to special use cases. However, if bad practices or implementation mistakes remain in the productive versions, adversaries may be able to perform man-in-the-middle (MiTM) attacks. If successful, such an attack allows intercepting and modifying all network data between the app and the bank's server.

Bad certificate chain checking (47%). Like hostname verification, this vulnerability also focuses on improper handling of the TLS handshake. Proper certificate chain checking involves verifying that the server's certificate is issued by a trusted CA, checking for certificate revocation status, and verifying the entire certificate chain up to a trusted root CA. However, if the app fails to check the certificate chain properly, it could potentially trust a certificate that has not been issued by a trusted certificate authority (CA) or that has been tampered with. As explained above, this could compromise the user's security via a man-in-the-middle attack.

WebView universal access (41%). The app configures a WebView such that JavaScript code loaded from files can access any other resource (regardless of its type), effectively completely disabling the same origin policy for code executed from a JS file. This configuration allows JavaScript code to access content providers and similar Android resources. Consequently, this vulnerability can lead to information disclosure from arbitrary resources.

Vulnrability for breach attack (41%). This is related to HTTP compression. If an app enables compression for an HTTP(s) connection when transferring sensitive data, an attacker can get information about the corresponding response by simply injecting plaintext into the request.

Bad protocol verifier (34%). The SSL/TLS protocol protects the app's data

by encrypting the connection between the client and server. If protocols considered insecure, such as SSL(v1), SSLv2, SSLv3, TLSv1, and TLSv1.1, are used, this can make connections vulnerable to exploits.

No SSL socket hostname verification (32%). SSL socket factory with hostname verification allows connections to be established by ensuring that the certificate presented by the server matches the expected hostname. If this is not done, it can lead to a vulnerability allowing MiTM attacks.

Insecure cryptography mode (32%). A cipher mode applies a cryptographic algorithm to encrypt or decrypt data larger than a single block. An insecure cipher mode, such as ECB, may enable attackers to decrypt data based on patterns that remain visible in the ciphertext. In banking apps, this may lead to severe data leaks.

Hard-coded cryptographic key (20%). A cryptographic key (**key**) is a secret that is used to encrypt or decrypt data (when using symmetric ciphers) or to generate signatures. If the Attacker knows the key, they can decrypt the data or forge signatures. Keys should never be hard-coded into apps because attackers can decompile the app to extract the key. Since all users have the same app with the same key, the Attacker may target all users and decrypt their data.

Furthermore, we have identified the components with the most critical SRCSs inside the app code to understand if these SRCSs stem from the app developer's code or libraries used by the apps with the help of AndroLibZoo [137]. As presented in Fig. 4.3, half of the apps have no SRCSs from their used libraries, i.e., all SRCSs stem from the original app code. Further, 75% of the apps have at least four SRCSs in the developer's code. The most critical SRCSes have been found in the developer's code.

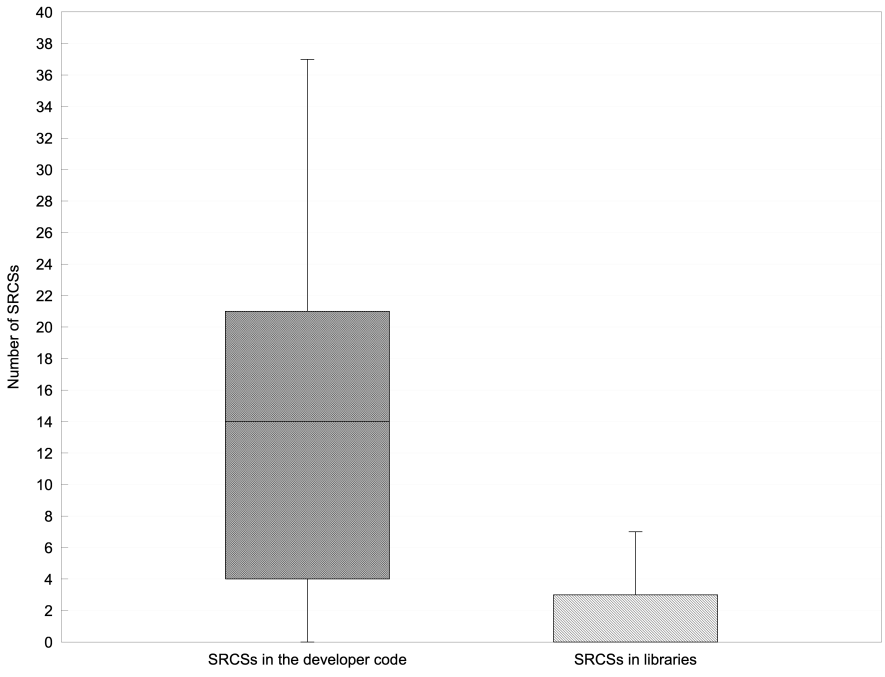


Figure 4.3: SRCSs in developer code and libraries.

To validate the results yielded by the scanner, we manually analyzed a representative sample of SRCSs in our set of MBAs. We confirm that 10 of the 21 SRCSs manually checked can be exploited; thus, they are vulnerabilities. Among those

10, 6 are among the top ten most common security issues. Table 4.1 shows these vulnerabilities and possible ways to exploit them.

Table 4.1: Description of vulnerabilities and possible exploit cases.

Security issues	Description	Possible way to exploit
Use of insecure cryptographic algorithm (CWE-327, CWE-328)	A cryptographic algorithm is used to encrypt and decrypt data by defining rules. Apps use insecure algorithms, such as SHA1, MD5, etc., in encrypting and decrypting sensitive data. Insecure use of cryptography may enable attackers to access sensitive information which is supposed to be protected. Using insecure algorithms can lead to reputation damage, data loss, etc.	By using dictionary attacks, an attacker can easily crack the hash using these hash functions.
Use of a bad hostname verify (CWE-295)	SSL/TLS HostnameVerify refers to the process of validating that the hostname of the server matches the hostname(s) listed in the X.509 certificate presented by the server during the SSL/TLS handshake. The SSL/TLS HostnameVerify process is not implemented correctly in some apps, allowing certificates with incorrect or mismatched hostnames to be accepted. This can lead to data breaches.	By using MiTM, attackers can spoof the SSL server via a valid certificate for a different host, leading to potential security breaches
Bad certificate chain checking (CWE-297)	Certificate chain checking ensures the validity of SSL certificates presented by the server when communicating with the app. The certificate must come from a trusted certificate authority (CA). Apps fail to check the certificate chain properly.	An attacker needs to be in a MiTM position to intercept the communication. When the victim uses the vulnerable app, the Attacker can intercept and modify the app and server communication.
No SSL socket hostname verification (CWE-297)	Apps create sockets to connect to specified hosts at specified ports. However, many do not perform a hostname verification when establishing connections.	Attacker can perform a MiTM attack to intercept communications and decrypt and modify data transferred for malicious purposes.
Insecure cryptography mode (CWE-327)	Apps use insecure cryptography cipher mode, such as ECB. Using insecure cipher modes may enable attackers to decrypt or tamper data, which can lead to serious security breaches and data leaks. With ECB, the same inputs result in the same ciphertext.	Since the same inputs result in the same ciphertext, an. An Attacker can use a chosen plaintext attack (CPA) to guess the encrypted data.
Hard-coded cryptographic key (CWE-321)	A cryptographic key (**key**) is used to generate a secret key from a SecretKeySpec method to encrypt and decrypt data. If "**key**" is known, attackers can deduce the secret key to decrypt data. Developers often hard-code this key into the app code.	By reverse-engineering the app, an attacker can get the hard-coded key and use it to decrypt any sensitive data that is stored or transmitted using the application or device.
Use of static initialization vector (CWE-1204)	The initialization vector is used to randomize cipher texts with the same input. When a static IV is used, the same inputs result in the same ciphertext, at least in parts. As such, a new IV should be selected randomly before each use. The IV is considered public information.	since the same inputs result in the same ciphertext when using static IV, an attacker can use a chosen plaintext attack (CPA) to guess the encrypted data.
Use of clear text password field (CWE-549)	To type confidential info such as password, pin code, etc., there are specific input types that allow hiding the information when typing, such as "textPassword", "numberPassword", etc. Instead of using these recommended input fields, many apps use clear text fields to enter this kind of sensitive info. This allows an adversary to observe the entry password, to be copied in the clipboard, or to be screen captured using malicious apps.	By developing malware, an attacker can capture password typing by the target, or by observing the entry password.
Sensitive data logged (CWE-532, CWE-534)	Apps log sensitive data such as username and password. Those logged data will be stored in the logcat memory.	Logged data can be accessed by privileged system apps, rooting or jailbreaking devices, or by developing mobile malware.
Hard-coded backend credentials (CWE-798)	For authentication to a backend server, the user must choose his/her credentials (username, password). In some apps, backend credentials are hard-coded for connection to a backend resource.	By reverse-engineering the app, an attacker can get those credentials and steal sensitive data from the backend resource.

Answer to RQ1

WAEMU banking apps present numerous security issues. Most of them have been found in the developers' code. Many of these security issues could be exploited by reverse engineering apps, developing malware, etc.

4.4.2 RQ2: How do security issues evolve in WAEMU bank apps?

This section describes the evolution of security issues through the various app versions. Among the 59 MBAs considered in this study, 47 have at least two versions, including the reference one. Recall that not all apps have the same number of

versions. To investigate the evolution, we only consider three versions for each app among those with at least three versions, representing 68% (32 out of 47 MBAs).

Figure 4.4 illustrates the evolution of sthe ecurity issues on the chosen versions.

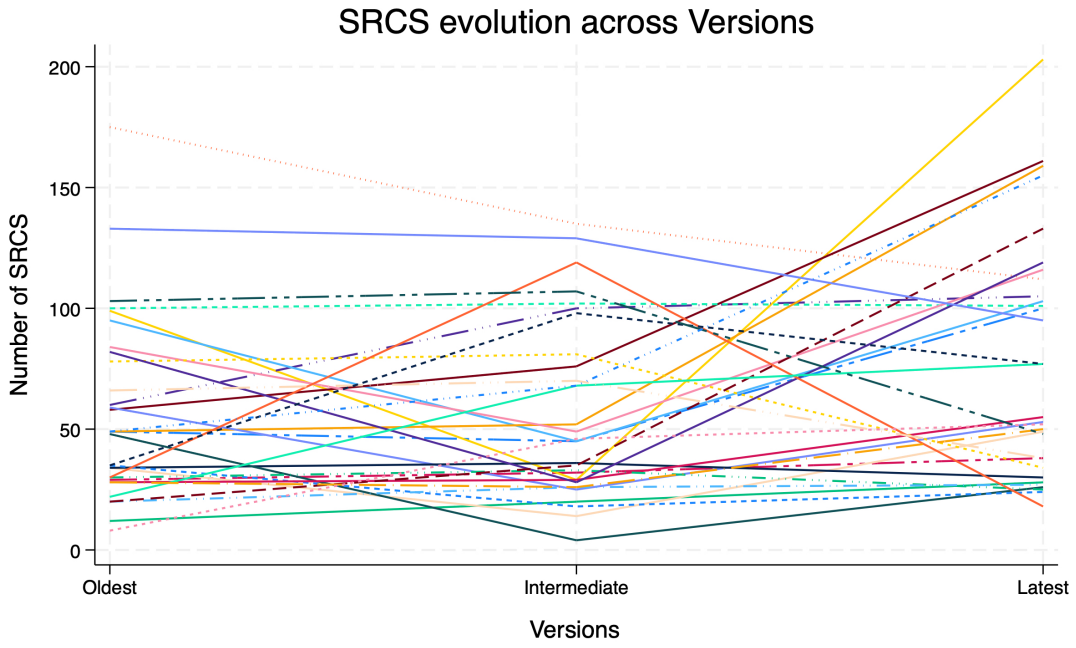


Figure 4.4: Security issue evolution across the app versions.

Around 20 (62%) out of the 32 MBAs have had increasing security issues between the first (oldest) and intermediate versions. This number is around 22 (68%) between the intermediate and the reference (latest) versions. Globally, approximately 21 (representing 65%) of these MBAs have had increasing security issues since the first version was available. However, some of them have seen a decreased number of security issues on the intermediate version, but it has increased significantly in the reference version.

To determine the percentage of security concerns that disappeared and the number of new ones, we considered all 47 apps. In Table 4.2, we give the means of the number and percentage of SRCSs that disappeared and the number of new ones between the versions. We can see that almost all the WAEMU banking apps have SRCSs disappeared, and many new ones appear from the first to the reference app.

Answer to RQ2

We observe an increasing trend of security issues with the update of apps for most of the studied MBAs. Indeed, while developers propose updates, the new versions are not necessarily more secure: some issues are fixed while new ones are introduced in the process.

Table 4.2: Tracking of the security issues across the app versions.

Apps	Average of disappear SRCSs (#)	Average of disappear SRCSs (%)	Average of new SRCSs (#)
A1	6.33	13.67	23.33
A2	2.25	7.35	4.5
A3	0	0	17
A4	2	5	16
A5	3.25	18.75	7.25
A6	41	40.2	46
A7	36.5	36.87	88.5
A8	29	25.22	48
A9	24.25	21.89	35.5
A10	2	22.22	3
A11	2	10	99
A12	67.67	48.43	46.67
A13	0	0	14
A14	2	5.88	7
A15	0	0	11
A16	2	22.22	2
A17	2	22.22	2
A18	34	37.87	36
A19	28	58.33	49
A20	42	35.28	67.75
A21	32	35.6	32.25
A22	12	24	14
A23	25	48.19	80
A24	14.5	27.82	13
A25	12.5	40.97	20
A26	22.25	31.92	134.25
A27	5.67	15.85	4.33
A28	17.75	24.14	25.75
A29	17.33	72.5	10
A30	0	0	16
A31	5.75	18.35	3
A32	3.5	12.5	17
A33	8.5	24.9	7.25
A34	0	0	0
A35	36.25	32.01	25.25
A36	24.33	38.84	36.67
A37	32	38.35	29
A38	0.75	2.96	2.5
A39	5	19.44	33.25
A40	5.75	10.9	19.5
A41	3	10.92	8.5
A42	56.75	48.85	47.25
A43	12	17.26	2.67
A44	28.75	52.23	39.25
A45	3.75	9.34	14.75
A46	18.5	18.74	4.75
A47	9	15.2	35.5

4.4.3 RQ3: How vulnerable are WAEMU banking apps compared to other banking apps?

General comparison. This comparison is based on all critical issues found in the apps. In other words, we take all the most critical issues from WAEMU banking apps and compare them with those coming from apps of the top 20 European Union (EU), top 20 United States (US), and top 20 other developing countries (ODC) banks. The results show that WAEMU banking apps are more secure than EU, US, and ODC banking apps. Indeed, as illustrated in Figure 4.5, an average of approximately 24 critical security issues per app is found in WAEMU banking apps. EU, US, and ODC banking apps present approximately 31, 39, and 38 critical security issues per app, respectively. We have also compared the security of those apps based on more

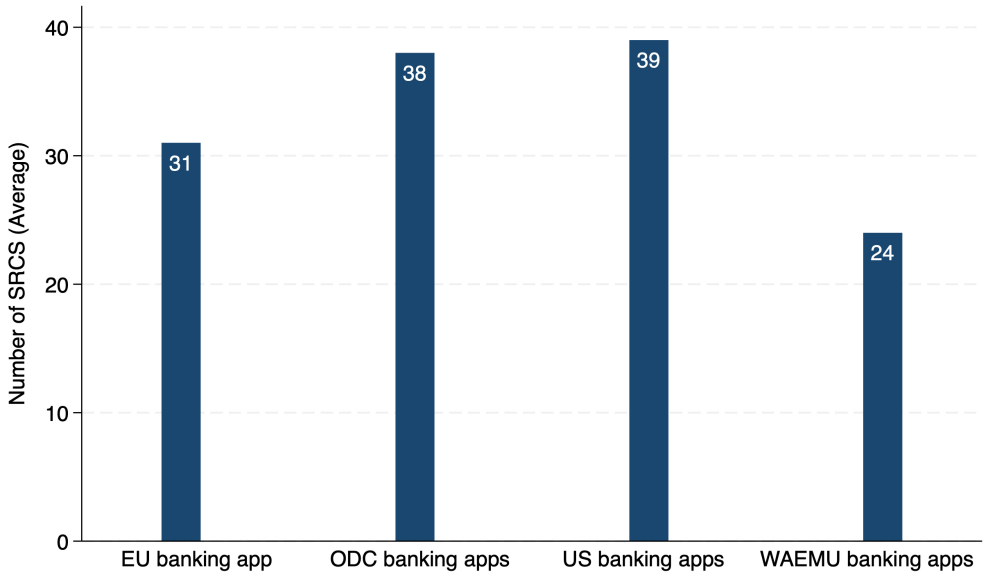


Figure 4.5: Comparison with WAEMU apps vs. EU, US, and other developing countries banking apps.

specific criteria, which we will explain next.

Comparison based on the security issue location. We compare the security of the MBAs based on the location of the security issues found. For EU banking apps compared to WAEMU banking apps, we found that around 75% of the EU apps have at least one critical issue found in libraries used against 50% of the apps from WAEMU, as illustrated in Figure 4.6. Furthermore, each of the EU apps has two or more critical issues in the developer code, unlike WAEMU apps, which have some that do not have any issues with the developer code. As shown in Figure 4.7, US banking apps have more issues found in libraries and the developer code than WAEMU banking apps. Indeed, Figure 4.7 shows that 50% of the US apps have more than ten critical issues in libraries, while only 50% of the WAEMU apps have critical issues found in libraries, and the number does not reach ten. In addition, each US banking app presents at least three critical issues in the developer code, whereas around 25% of the WAEMU apps have less than four critical issues. As for MBAs from other developing countries, Fig. 4.8 shows that around 75% of them present a critical issue found in libraries. In addition to this, around 50% of them have at least

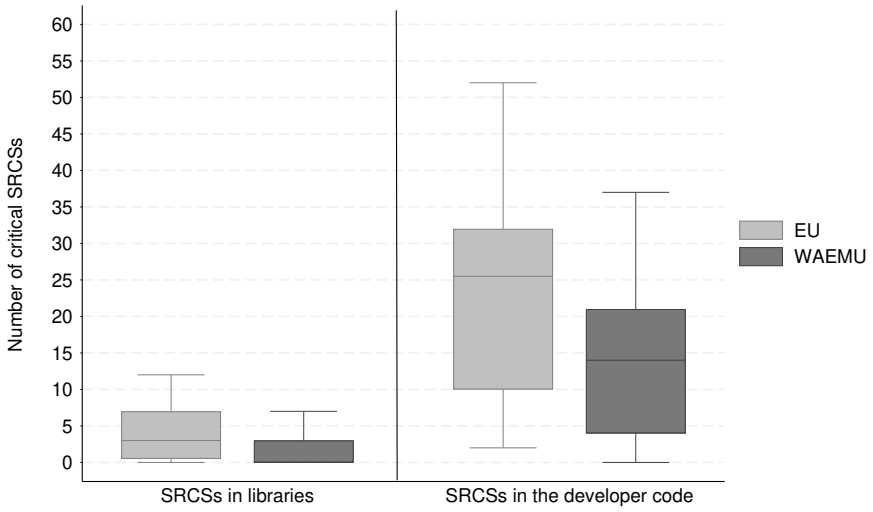


Figure 4.6: Comparison with WAEMU vs. EU banking apps based on the security issue location.

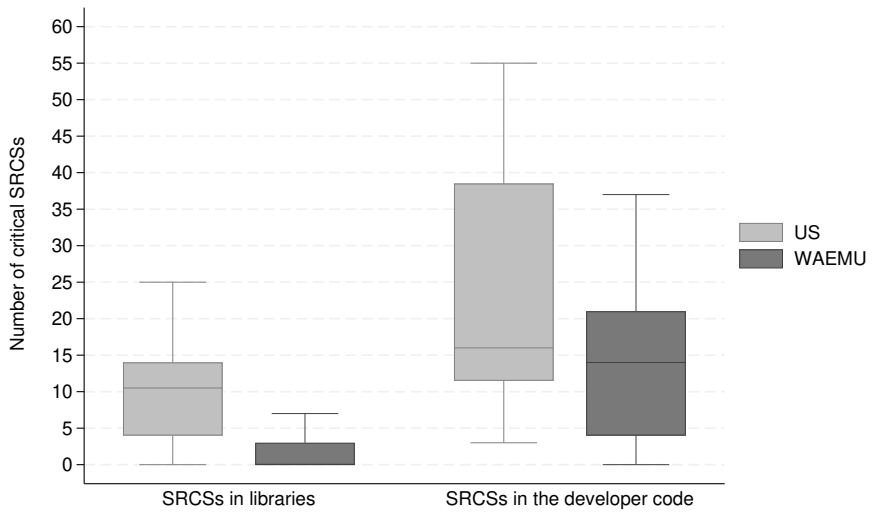


Figure 4.7: Comparison with WAEMU vs. US banking apps based on the security issue location.

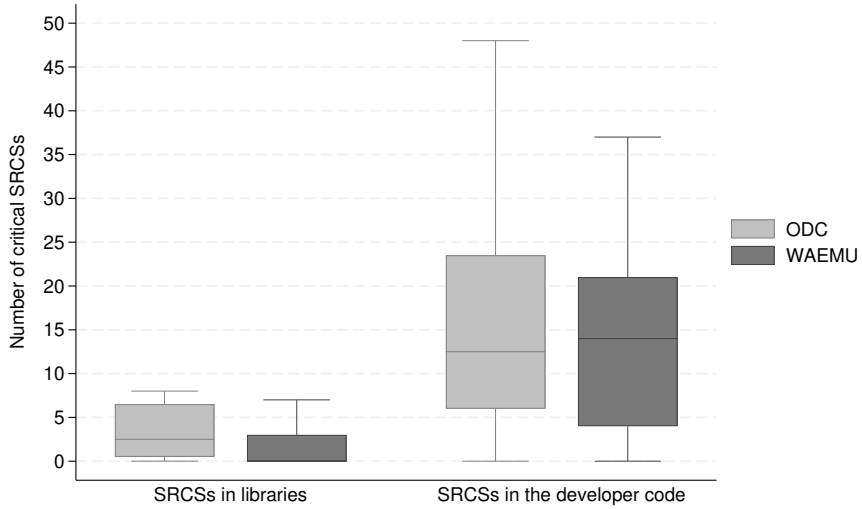


Figure 4.8: Comparison with WAEMU apps vs. other developing countries' banking apps based on the security issue location.

two critical issues, whereas only 50% of WAEMU apps present at least one critical issue found in libraries. Besides that, Fig. 4.8 also shows that 50% of the MBAs from other developing countries present at least 12 critical issues in developer code compared to WAEMU apps, in which 50% present 14 critical issues in the developer code.

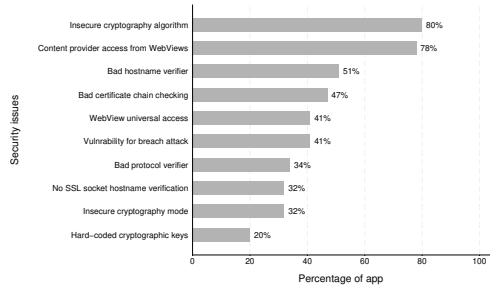
Comparison based on the top ten most common critical issues. Most of the top security issues in the EU, US, and ODC are the same in WAEMU, as highlighted in Figure 4.9, including static initialization vector (IV), hard-coded password, bad SSL error handling, and file system access through WebView.

A random initialization vector (IV) ensures that encrypting the same data multiple times leads to different ciphertexts. This precaution prevents attackers from matching ciphertexts against pre-computed tables (e.g., Rainbow Tables) to derive the plaintext. If the IV is constant, this protection is rendered ineffective, and tables can be pre-computed for this IV. This problem is common in EU, US, and ODC banking apps as illustrated in Figure 4.9b, 4.9c, and 4.9d.

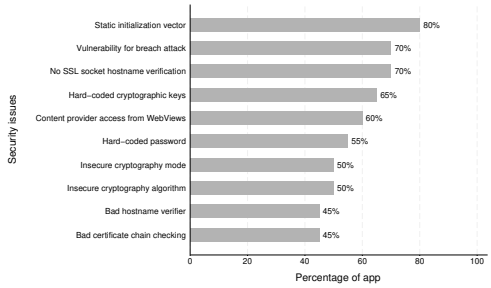
In WAEMU apps, on the other hand, most apps use old algorithms for encrypting and decrypting data, as highlighted by the high number of apps containing insecure cryptography algorithm issues in Figure 4.9a. Such algorithms can be broken regardless of a potential IV problem.

At least half of the EU, US, and ODC banking apps use hard-coded passwords to derive a cryptography key, which is then used to encrypt and decrypt data. Such behavior is more complex than directly hard-coding the key, but leads to the same vulnerability. In 20% of the WAEMU apps, this key is directly hard-coded, avoiding the key derivation step. In either case, attackers can reconstruct the key and decrypt the data.

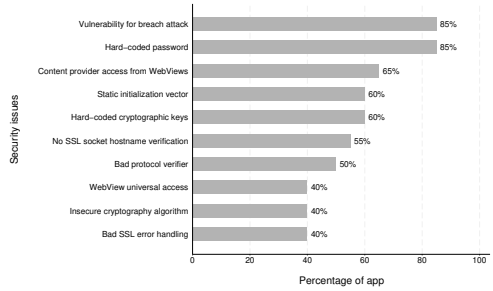
An SSL error can occur when a client loads a resource through a WebView. Consequently, the applications should properly handle SSL errors, allowing SSL connections to be aborted when errors occur. In US and ODC apps, as presented by their top ten in Figure 4.9c and 4.9d, many apps ignore SSL certificate errors, making them vulnerable to man-in-the-middle attacks. WAEMU apps, on the other



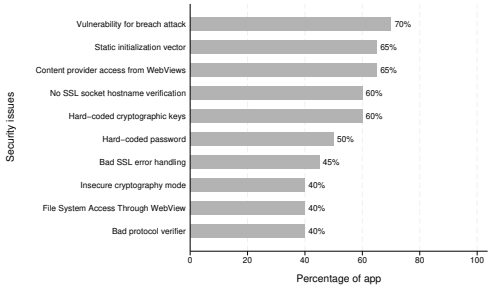
(a) Top ten of WAEMU apps' most common critical issues.



(b) Top ten of EU apps' most common critical issues.



(c) Top ten of US apps' most common critical issues.



(d) Top ten of ODC apps' most common critical issues.

Figure 4.9: Comparison of WAEMU and EU, US, and other developing country banking apps.

hand, do not have these issues in their top ten.

As highlighted by the percentage in Figure 4.9d, many ODC apps configure a WebView such that the WebView can only access files on the device's file system. In WAEMU apps, this practice is uncommon. However, they use universal access, which allows access to any resource, including files.

Answer to RQ3

Banking apps from the WAEMU appear to present fewer critical issues than MBAs from the top 20 of the EU, the US, and other developing countries (ODC). Their security issues, as well as those of the apps from the EU, the US, and other developing countries, come from libraries and developers' code. However, WAEMU apps present fewer critical issues in the developer code, as well as in libraries. Based on the top 10 security issues, WAEMU apps have avoided some of the critical issues that others have, but some specific issues are more prevalent.

4.4.4 RQ4: How vulnerable are child banking apps compared to parent banking apps?

Some of the WAEMU financial institutions are subsidiaries (child banks) of banks from other regions (parent banks). Some parent banks are from African countries (African parent banks), and others are from non-African countries (International

parent banks). In this section, we perform a security app comparison of child banking apps with their respective parent banking apps from both regions.

On the one hand, 13 out of 59 MBAs are African child banking apps. As illustrated in Figure 4.10, all of them inherit security issues from African parent banking apps, representing a mean of 37% security issues inherited. Furthermore, at least 17% of their security issues are new ones. On the other hand, 6 out of 59

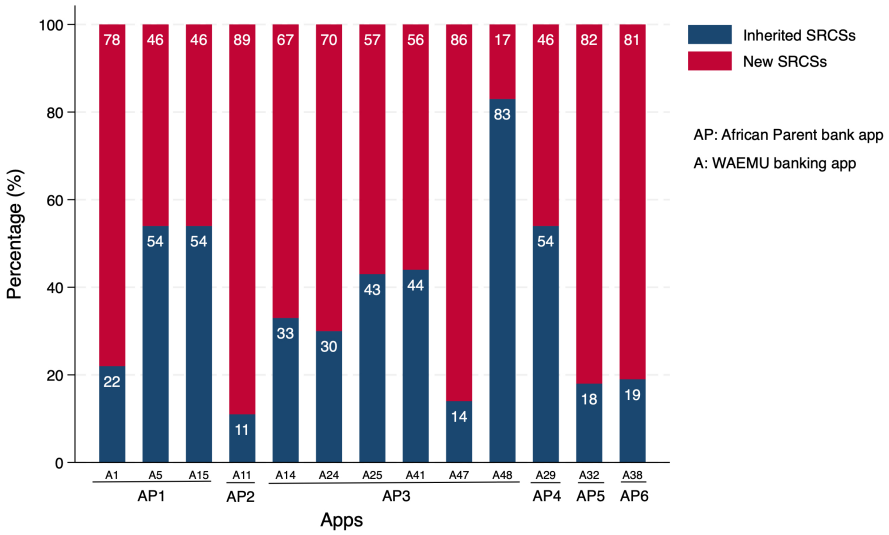


Figure 4.10: Comparison with WAEMU apps vs. other developing countries' banking apps based on the security issue location.

MBAs are International child banking apps. As illustrated in Figure 4.11, all of the International child banking apps inherit security issues from International parent banking apps, representing a mean of 20% security issues inherited. More than 67% of their security issues are new.

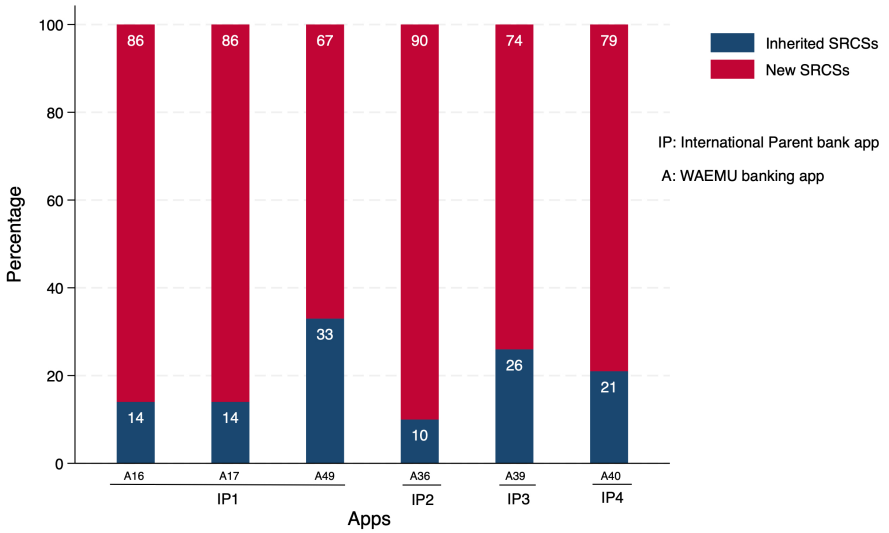


Figure 4.11: Comparison with WAEMU apps vs. other developing countries' banking apps based on the security issue location.

Answer to RQ4

Child apps always inherit security issues from the parent apps, with a significant number. Furthermore, they consistently introduce several new ones.

4.5 Discussion

4.5.1 Discussion about results

Our analysis reveals many SRCSs in the developers' code and the libraries used by the WAEMU banking apps. However, most of them have been found in the developers' code, suggesting that developers compromise, intentionally or not, the security of users. To avoid such security issues, developers must understand the best practices and know how the required features of a banking app can be implemented securely.

Based on the most critical security issues (csi) found in the MBAs, WAEMU banking apps present fewer issues (average of 24 csi/app approximately) than EU, US, and ODC banking apps, which show 31, 39, and 38 csi/app, respectively. Most of these security issues have been found in the developer code. On average, WAEMU apps have fewer issues than EU, US, and ODC banking apps introduced by developers. This could be explained by the higher number of lines of code in non-WAEMU banking applications than in WAEMU banking applications. It could simply mean that WAEMU banking application developers are more concerned with best practices when developing applications than non-WAEMU banking application developers.

Among the security issues found, many could be potentially exploitable. For instance, cryptography is used to secure users' sensitive data and avoid unauthorized access by attackers [138]. However, many cryptographic algorithms are considered outdated and insecure over time, including SHA1 and MD5. However, they are still used prevalently in mobile banking apps. As illustrated in Listing 4, apps often use insecure hash functions to hash sensitive data. In this app, when updating the password, the new one is recuperated (*in line 4*), hashed using MD5, and put in a string variable (*in line 32*). When a malicious person retrieves this string value, the password can be easily recovered. This practice is the most common since it affects 80% of the WAEMU apps. This suggests that all the affected apps could be potentially compromised, leading to significant impacts, including data breaches, data loss, and reputation damage [139]. It is also essential to use strong cryptographic algorithms. Still, it is also crucial for developers to check the best practices regularly since algorithms considered more secure could be insecure over time. Besides that, banking apps often use a hard-coded cryptographic key to encrypt and decrypt data. If this key is known by a malicious person (which is trivial to achieve by decompiling the app), it significantly increases the possibility of recovering the encrypted data [140]. The code snippet in Listing 5 shows an example of such a hard-coded key (*line 4*) being used for generating a secret key (*line 9*) which is used in the **encode** (*line 12*) and **decrypt** (*line 19*) methods. These methods are later used for encrypting and decrypting data, including passwords, one-time passwords (OTP), and secret questions/answers. Our analysis found this in 20% of the apps, representing the 10^{th} most common security issues. There are more vulnerabilities

```
1   public class classeName {
2       [...]
3       final EditText editText = (EditText) findViewById(R.id.txtOldPassword)Al
4       final EditText editText2 = (EditText) findViewById(R.id.txtNewPassword) ;
5       final Editrext editiexts = (Edittext) rindviewbyid(K.d.txtcontirmpassword);
6       ((Button) findViewById(R.id.btnChange)).setOnClickListener(new View.OnClickListener) {
7       @Override
8       public void onClick(View view) {
9           final String str2;
10          if (...) {
11          [...]
12          }
13          else {
14              String obj = editText2.getText(). toString();
15              try {
16                  MessageDigest instance = MessageDigest.get Instance ("MD5");
17                  instance.update("P@#$&*(-+"-getBytes(), 0, 10);
18                  String str3 = obj * new Biginteger(1, instance.digest).toString (16);
19                  MessageDigest instance2 = MessageDigest.getInstance ("MD5");
20                  instance2.update(str3.getBytes(), 0, str3.length());
21                  obj = new BigInteger(1, instance2.digest()).toString(16);
22              } catch (Exception unused) {
23              }
24              [...]
25              String obj2 = editText2.getText).toString();
26              try {
27                  MessageDigest instance3 = MessageDigest-getInstance ("MD5");
28                  instance3.update("P@#$&*(-+"-getBytes(), 0, 10);
29                  obj2 = obj2 + new BigInteger(1, instance3.digest()).toString (16);
30                  MessageDigest instance4 = MessageDigest.getinstance ("MD5");
31                  instance4.update(obj2.getBytes(), 0, obj2. length()):
32                  str2 = new BigInteger(1, instance4.digest()).toString(16);
33              } catch (Exception unused2) {
34                    str2 = obj2;
35              }
36              [...]
37          }
38          [...]
39      }
40      [...]
41  }
```

Listing 4: App using an insecure crypto algorithm.

among the ten most prevalent security issues such as access to content providers within WebView (78%), which may allow access to protected content [141], bad hostname verifier (51%), which may enable certificates with incorrect or mismatched hostnames to be accepted leading to data breaches, bad certificate chain checking (47%), which could potentially trust a certificate that has not been issued by a trusted certificate authority or that has been tampered with allowing a man-in-the-middle attack, no SSL socket hostname verification (32%) when establishing connections, which can lead to a man-in-the-middle attack, and more. Several MBAs have the same developer, which could explain the frequency of specific security problems.

Even outside the top 10 list, there are other severe vulnerabilities that, albeit not very common, can be potentially exploitable and affect several apps. For example, several WAEMU apps log sensitive information such as the user's password under some circumstances, as illustrated at *line 26* in listing 6. This vulnerability could be exploited by simply connecting the device to a PC and using adb[3] commands [142]. Other apps installed on the same device, such as malicious or privileged system apps, could potentially exploit this vulnerability.

Developers should avoid bad practices, including hard-coded keys and credentials, and logging sensitive data. They must properly verify the hostname and certificate authority before connecting with servers through the apps.

By investigating security issue evolution through the app versions, we identified an increased number between the first and the intermediate versions in most WAEMU apps (approximately 62%), as well as between the intermediate and the

[3] `https://developer.android.com/tools/adb`

```
1   public class globalmethods {
2       private static final String ALGO = "AES";
3       [...]
4       private static final byte[] keyValue = {64, 97, 112, 101, 120, 53, 48, 102, 116, 119, 64, 114, 51, 53, 49, 56};
5
6       [...]
7
8       private static Key generateKey() throws Exception {
9           return new SecretKeySpec(keyValue, ALGO);
10      }
11
12      public static String encode(String str) throws Exception {
13          Key generateKey = generateKey();
14          Cipher instance = Cipher.getInstance(ALGO);
15          instance.init(1, generateKey);
16          return Base64.encodeToString(instance.doFinal(str.getBytes()), 0).replace(IOUtils.LINE_SEPARATOR_UNIX, "");
17      }
18
19      public String decrypt(String str) {
20          try {
21              Key generateKey = generateKey();
22              Cipher instance = Cipher.getInstance(ALGO);
23              instance.init(2, generateKey);
24              return new String(instance.doFinal(Base64.decode(str.getBytes(), 0)));
25          } catch (Exception unused) {
26              return str;
27          }
28      }
```

Listing 5: App using a hard-coded key.

reference versions (approximately 68% This indicates that even if banks and financial institutions regularly propose updates (at least three versions in 68% apps), these updates are ineffective in fixing the security issues. Ideally, updates should improve the security of the apps by fixing security issues. Based on the numbers presented in Table 4.2, several SRCSs have disappeared between the first and the last (reference) version. However, the number of new SRCSs has risen significantly, almost doubling the number of disappeared SRCSs in most WAEMU apps.

We also noticed that several banks and financial institutions do not propose updates regularly. For instance, in more than one year, they could propose a single update or not. This could suggest they do not care about their customers' security. Apps should be regularly updated to consider new security trends, and issues must be fixed after updates.

Of the 59 WAEMU banking apps, 19 are from banks with parents in a non-WAEMU country, among which 13 have African parents (AP), while 6 have international parents (IP). We noticed interesting findings by comparing these apps with those from the parent banks. Indeed, the 19 child banking apps inherit security issues from parent banking apps: from 11 to 83% of the issues for AP and from 14 to 33% for IP. It has also been noticed that even if all the child banking apps introduce several new security issues, most parent banking apps have more security issues than the child banking apps.

4.5.2 Limitations and future directions

This work, investigating the security of mobile banking apps in WAEMU countries, highlights the security issues introduced mainly by the developers. In this study, we point out potentially exploitable security issues and propose use cases that could exploit them. As it is essential to know if these security issues are exploitable to understand better the security impacts, the future direction could be to propose approaches and methods for exploiting these issues. When comparing the security of WAEMU apps with those from other regions, we focused solely on the total

```
1   public class CodeSecretActivity extends AppCompatActivity {
2       private Button btnConfirmer;
3       private EditText editTextCodeSecret;
4       private EditText editTextNewCodeSecret;
5       private Intent passwordIntent;
6       private MyProgressDialog pbar;
7       @Override
8       public void onCreate(Bundle bundle) {
9           new ProgressUpdatePassword().execute(stringExtra, stringExtra2,
            ↪  CodeSecretActivity.this.editTextNewCodeSecret.getText().toString());
10          this.editTextCodeSecret = (EditText) findViewById(R.id.edittext_code_login);
11          this.editTextNewCodeSecret = (EditText) findViewById(R.id.edittext_new_code_login);
12          this.btnConfirmer = (Button) findViewById(R.id.btn_new_code);
13          Intent intent = getIntent();
14          this.passwordIntent = intent;
15          final String stringExtra = intent.getStringExtra("username");
16          final String stringExtra2 = this.passwordIntent.getStringExtra("password");
17          this.btnConfirmer.setOnClickListener(new View.OnClickListener() {
18              @Override
19              public void onClick(View view) {
20                  if (...) {
21                      [...]
22                  } else {
23                      new ProgressUpdatePassword().execute(stringExtra, stringExtra2,
                        ↪  CodeSecretActivity.this.editTextNewCodeSecret.getText() .toString());
24                      Log.i("username ", stringExtra);
25                      Log.i("password ", stringExtra2);
26                      Log.i("new password ", CodeSecretActivity.this.editTextNewCodeSecret.getText() .toString());
27                  }
28              }
29          });
30      }
31      [...]
32  }
```

Listing 6: App logging a password.

number of security issues, the ten most common issues, and their locations. However, relying on the number of security issues alone can introduce bias, as the number of issues may correlate with the app's size or the number of lines of code. To address this, future investigations could consider the number of lines of code in each app to achieve more accurate and unbiased results. Mobile money apps are as crucial as mobile banking apps since they contribute to financial inclusion. Future research could investigate those apps and extend the study context rather than limiting it to WAEMU countries only.

4.5.3 Threat to validity

We gathered the latest app versions from WAEMU banks in December 2022. It is likely that from then until this paper's composition, banks have introduced new APK versions by addressing and rectifying security issues. This could potentially impact the relevance of this study's findings.

We have manually reviewed the app source code to verify the identified SRCSs and determine which ones pose vulnerabilities. However, due to time limitations and the abundance of certain SRCSs, we could not examine all SRCSs in every MBA. Understanding that an SRCS might be a vulnerability in some apps but not others is crucial. For instance, a scanner might detect SHA1 or MD5 hashing sensitive data such as usernames and passwords in one app. In contrast, in another app, it might hash data that does not significantly compromise user security.

We examined the evolution of security issues across app versions, focusing only on those with at least three versions. We tracked the occurrence of each SRCS within these apps, comparing its frequency between consecutive versions. If an SRCS's frequency increased in the next version, we considered it as 'increased', with the increase being the difference in occurrence numbers between the two versions. If not, we considered the SRCS as 'fixed'. Thus, we considered an SRCS 'fixed' if its

occurrence number decreased in the subsequent version. It is important to note that an issue might disappear due to intentional fixing or code snippet deletion. An SRCS instance might be present in one version and disappear in the next, regardless of whether its occurrence number has surged. Conversely, new SRCSs might emerge even if the occurrence number has dropped. Given the difficulty in tracking disappeared SRCS instances, we focused solely on occurrence numbers.

4.6 Summary

This work performed a security assessment of mobile apps from banks and financial institutions in the West African Economic and Monetary Union (WAEMU) countries. We have performed a static analysis of fifty-nine (59) collected mobile banking apps (MBAs) from the eight WAEMU countries, and the results show that many vulnerabilities can lead to several real-world attacks.

An attacker could reverse-engineer the apps to retrieve hard-coded backend credentials and cryptographic keys for encrypting/decrypting data. Additionally, attackers could use dictionary attacks to crack hashes obtained from insecure algorithms such as SHA1 and MD5. They could develop mobile malware to extract sensitive data, such as usernames and passwords, from log files in the device or capture credentials entered into unprotected (clear text) UI fields. Most security issues are found in the developers' code. This indicates that developers, intentionally or not, introduce vulnerabilities into mobile apps, resulting in user privacy violations and potential damages. As a result, the reputation of financial institutions can be compromised.

Some banks and financial institutions offer updates to their mobile banking apps (MBAs), but these updates often do not address all security issues. Additionally, new security vulnerabilities frequently emerge in the latest versions. Other institutions either do not provide updates or do so irregularly. The emergence of new threats and exploitation techniques can compromise app security if the latest protection techniques are not implemented and newly discovered security breaches are not addressed.

According to the results of the comparison of WAEMU apps with European Union (EU) apps, United States (US) apps, and developing countries other than WAEMU countries (ODC), MBAs from WAEMU seem to be more secure than those from different countries. Some WAEMU apps are banking apps of subsidiaries of banks in other regions. Our analysis shows that those apps always inherit some security issues of their parent apps, and almost all present several new ones. This indicates that subsidiary apps seem more vulnerable and can compromise user security much more than parent apps.

As part of responsible disclosure, we have contacted some banks to discuss our findings regarding their MBAs. The goal was to help them fix these concerns to improve the security of their apps. It was hard to find a security contact for reporting in many banks. In some cases, the email address in the Play Store did not exist or was a Gmail address. We did not get responses from the banks we tried to contact. No bank had a proper vulnerability disclosure process.

We have givent recommendation to the developers to allow them to avoid some practices that can lead to damage and adopt more secure practices.

5

Conclusion and Future works

In this chapter, we conclude this dissertation with an overview of our key contributions and enumerates relevant research directions for future work.

Over the past decade, mobile technologies have deeply transformed the digital landscape worldwide. Smartphones now have become the primary means of accessing online services for billions of people. In developing countries, this transformation is even more striking. Mobile phones become the primary and often represent the only connection to the Internet, enabling access to education, healthcare, communication, and financial services. As a result, the smartphone has evolved from a simple communication tool into an essential instrument of social and economic participation. Yet, this diffusion of mobile technology has unfolded in an environment characterized by low-cost devices, fragmented software ecosystems, and limited regulatory supervision. In such contexts, the promise of digital inclusion often coexists with invisible security risks. This thesis explores least examined aspects of this digital landscape identified after an empirical evidence from a systematic literature review: the security and privacy implications of pre-installed applications and mobile banking apps in developing countries.

The systematic review reveals a few, yet focused, bodies of research on mobile app security in developing countries, with vulnerability detection and FinTech apps dominating the literature, highlighting an overall need for more specialized techniques and datasets tailored to these contexts. Field analyses of real devices and apps complete that picture with concrete findings: a sizable fraction of pre-installed/vendor apps on low-cost African smartphones leak sensitive information, expose sensitive components, execute dangerous commands, interact with SMS, and sometimes perform silent installations, demonstrating that the device supply chain and vendor software are important attack surfaces. Similarly, an assessment of mobile banking apps across the WAEMU region reveals that insecure development practices introduce exploitable issues that often persist across updates. While these apps may exhibit fewer critical issues than some international peers, the remaining weaknesses still pose risks to users and financial institutions. Taken together, these works argue that the security problem is multidimensional, encompassing developer practices, vendor/ODM supply chains, app update and maintenance cycles, and user-facing deployment contexts (characterized by limited infrastructure and lower digital literacy). Addressing this, therefore, requires a combination of tailored static/dynamic analysis tools, better secure-by-design development practices for financial and high-risk applications, greater scrutiny of vendor firmware and pre-installed packages, targeted benchmarks and datasets for the region, and policy/industry engagement to improve the auditability of the supply chain.

In sum, the literature review and empirical evaluations converge on the conclusion that mobile security in developing regions is an under-researched but urgent problem: practical improvements demand region-aware datasets and tools, stronger supply-chain supervision for low-cost devices, and the adoption of secure development and maintenance practices by app developers and financial institutions, only through coordinated technical, organizational, and policy efforts can the persistent risks identified in these studies be significantly reduced.

As future works, we have identified several topics that we want to explore:

- **Analysis of pre-installed apps (Extension).** We have performed a study on Android devices primarily used in Africa by statistically analysing the applications they ship. This study has identified suspicious apps by focusing on the most common behaviors malware regularly implements. It has also detected

apps that exfiltrate sensitive information through exported components and other means. We plan to add additional modules and enhance the current ones to provide a more comprehensive and robust detection pipeline. The pipeline will provide a dashboard with comprehensive details for each application concerning its security. Additionally, we will integrate a module that proposes a dynamic analysis of the applications using a large language model. Furthermore, we plan to increase the number of devices to analyze to cover a wide range of models from the targeted brands.

- **An in-depth analysis of mobile VPN apps used in Africa.** In several African countries, governments periodically restrict access to the Internet or to specific online services, prompting populations to rely on mobile Virtual Private Network (VPN) applications to bypass these controls. Due to economic constraints and ease of access, free VPN apps are widely adopted, often under the assumption that they provide privacy and security. However, the security implications of using free mobile VPNs remain largely unclear, as their business models are opaque and their privileged position on mobile devices allows them to intercept all network traffic, potentially enabling extensive data collection, tracking, or abuse. In contexts where smartphones are the primary means of Internet access and are used for sensitive services such as mobile banking and digital identity, these risks are particularly critical. This work is motivated by the lack of empirical evidence on the real behavior of mobile VPN apps used in Africa and aims to systematically assess their security and privacy properties through an in-depth analysis of their permissions, network activity, and data handling practices.
- **Investigating malware targeting developing countries.** Our SLR has revealed unexplored research directions, including the analysis and identification of malware specifically targeting developing countries. In 2022, among the ten countries most targeted by malware, the majority are developing countries [25]. This means that mobile malware in developing countries needs more interest from researchers. Unfortunately, there is no study focusing on proposing an approach or methodology tailored to the specific context of these regions to identify malware. One of our future projects will be to identify the types of malware used, their payloads, techniques, and procedures employed by attackers to distribute and compromise devices, as well as the active threat actors and information assets that are most frequently targeted.
- **Analysis of USSD technology used on Mobile Money.** Mobile money is widely used in developing countries. Its deployment is often based on USSD (Unstructured Supplementary Service Data) technology because it works on every phone, does not require data, and is extremely simple to use. USSD has become the backbone of financial inclusion for millions of people in developing countries. Even though USSD plays a central role in national digital economies, it relies on old signalling protocols, minimal encryption, and no strong authentication. And while fintech services have become more sophisticated, USSD's core model has remained largely unchanged. However, it brings several and distinct threats, including SIM swap, session hijack, malware overlays, social engineering, fake USSD menus, weak session management, and more. As future work, we plan to explore this topic to contribute to enhancing the security of the mobile money infrastructure.

List of Papers

Papers included in this dissertation:

- DIALLO, A., SAMHI, J., BISSYANDE, T. F., and KLEIN, J., (In)Security of mobile apps in developing countries: a systematic literature review. Empir Software Eng 30, 131 (2025).
- DIALLO, A., DIOP, A., KABORE, A. K., PILGUN, A., SAMHI, J., BISSYANDE, T. F., and KLEIN, J., On the security of pre-installed Android apps in low-cost devices. 17th EAI International Conference on Africa Internet Infrastructure and Services (EAI AFRICOMM) (2025).
- DIALLO, A., WAR, A., DIOUF, M. A., SAMHI, J., ARZT, S., BISSYANDE, T. F., and KLEIN, J., "Security Assessment of Mobile Banking Apps in West African Economic and Monetary Union," 2025 Cybersecurity4D (C4D), Gqeberha, South Africa, 2025, pp. 1-13.

Papers not included in this dissertation:

- WAR, A., DIALLO, A., HABIB, A., KLEIN, J., and BISSYANDE, T. F., Vulnerabilities in infrastructure as code: what, how many, and who?. Empir Software Eng 30, 120 (2025).
- MBODJI, F. N., SOUGOUFARA, M. M. C., OUEDRAOGO, W. A. M. C., DIALLO, A., LIU, K., KLEIN, J., and BISSYANDE, T. F., (2025). evalSmarT: An LLM-Based Framework for Evaluating Smart Contract Generated Comments. In The 40th IEEE/ACM International Conference on Automated Software Engineering, ASE 2025. Seoul, South Korea: IEEE/ACM.
- MBODJI, F. N., NIANG, A. L., DIALLO, A., KLEIN, J., and BISSYANDE, T. (2025). Affordance–Experimentation–Actualization- Evolution for AI-based Support of Innovation. In Reimagine the Future Through the Advancement of Information Technology. Polokwane, South Africa: IEEE.

Bibliography

[1] Statcounter, “Mobile operating system market share worldwide,” https://gs.statcounter.com, accessed: 2025-12-20.

[2] A. Documentation, “Product partitions,” https://source.android.com/docs/core/architecture/partitions, accessed: 2025-12-30.

[3] J. A. Olson, D. A. Sandra, É. S. Colucci, A. Al Bikaii, D. Chmoulevitch, J. Nahas, A. Raz, and S. P. Veissière, “Smartphone addiction is increasing across the world: A meta-analysis of 24 countries,” *Computers in Human Behavior*, vol. 129, p. 107138, 2022.

[4] A. Turner, “How many smartphones are in the world?” https://www.bankmycell.com/blog/how-many-phones-are-in-the-world#1579705085743-b3697bdb-9a8f, accessed: 2025-10-16.

[5] OSIRIS, “En afrique subsaharienne, le taux d’adoption des smartphones atteindra 87% en 2030,” https://www.osiris.sn/En-Afrique-subsaharienne-le-taux-d.html, accessed: 2025-10-16.

[6] kaspersky, “Malware attacks in africa are increasing, reaching 85 million in only 6 months,” https://kaspersky.africa-newsroom.com/press/malware-attacks-in-africa-are-increasing-reaching-85-million-in-only-6-months?lang=en, accessed: 2025-10-16.

[7] TRUSTONIC, “Closing the digital divide with android smartphones in africa,” https://www.trustonic.com/opinion/closing-the-digital-divide-with-android-smartphones-in-africa/, accessed: 2025-10-16.

[8] A. Ng, “Android malware that comes preinstalled is a massive threat,” https://www.cnet.com/tech/mobile/android-malware-that-comes-preinstalled-are-a-massive-threat/, accessed: 2025-10-16.

[9] OSIRIS, “Potentially millions of android tvs and phones come with malware preinstalled,” https://arstechnica.com/information-technology/2023/05/potentially-millions-of-android-tvs-and-phones-come-with-malware-preinstalled/, accessed: 2025-10-16.

[10] Google, “Android (Go edition),” https://developer.android.com/guide/topics/androidgo, accessed: 2025-08-26.

[11] J. BARTON, “Orange releasing android-powered ultra-low cost smartphone for africa,” https://developingtelecoms.com/telecom-technology/telecom-devic

es-platforms/10053-orange-releasing-android-powered-ultra-low-cost-smart phone-for-africa.html, accessed: 2025-10-16.

[12] B.E.I. (2016) Le secteur bancaire en afrique subsaharienne: Évolutions récentes et inclusion financière numérique. [Online]. Available: https: //www.eib.org/attachments/efs/economic_report_banking_africa_digital _financial_inclusion_fr.pdf

[13] V. Lazović and T. Duričković, "The digital economy in developing countries-challenges and opportunities," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014, pp. 1580–1585.

[14] W. P. Review. (2024) Literacy rate by country 2024. [Online]. Available: https: //worldpopulationreview.com/country-rankings/literacy-rate-by-country

[15] approov. (2023) Security challenges of financial mobile apps in africa. [Online]. Available: https://approov.io/info/security-challenges-of-financial-mobile-a pps-in-africa

[16] F. Osman, W. H. Hassan2, Y. Yamada, and S. Goudarzi4, "Challenges of mobile money to mobile transaction service on policy regulation and security frauds in east africa," in *ASIA International Multidisciplinary Conference 2017*, 2017.

[17] S. Castle, F. Pervaiz, G. Weld, F. Roesner, and R. Anderson, "Let's talk money: Evaluating the security challenges of mobile money in the developing world," ser. ACM DEV '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/3001913.3001919

[18] A. Diallo, J. Samhi, T. F. Bissyandé, and J. Klein, "(in)security of mobile apps in developing countries: a systematic literature review," *Empirical Software Engineering*, vol. 30, no. 5, p. 131, Jul 2025. [Online]. Available: https://doi.org/10.1007/s10664-025-10689-z

[19] A. Turner, "How many apps are there in the world," https://www.bankmycell .com/blog/number-of-mobile-apps-worldwide, accessed: 2025-10-16.

[20] W. P. Review, "Developing countries 2023," https://worldpopulationreview.co m/country-rankings/developing-countries, accessed: 2025-10-16.

[21] Keyideas, "Impact of smartphones over society," https://www.keyideasinfote ch.com/blog/impact-of-smartphone-on-society/, accessed: 2025-10-16.

[22] R. Sheldon, "Advantages and disadvantages of mobile devices in business," https://www.techtarget.com/searchmobilecomputing/feature/Discover-the -benefits-of-mobile-devices-in-the-enterprise, accessed: 2025-10-16.

[23] A. Alaiad, M. Alsharo, and Y. Alnsour, "The determinants of m-health adoption in developing countries: an empirical investigation," *Applied clinical informatics*, vol. 10, no. 05, pp. 820–840, 2019.

[24] I. Pentina, L. Zhang, H. Bata, and Y. Chen, "Exploring privacy paradox in information-sensitive mobile app adoption: A cross-cultural comparison," *Computers in Human Behavior*, vol. 65, pp. 409–419, 2016.

[25] Statica, "Percentage of mobile users who have fallen victim to mobile malware infections in 3rd quarter 2022, by country," https://www.statista.com/statistics/325201/countries-share-of-malicious-attacks/, accessed: 2025-10-16.

[26] W. Carter, "Forces shaping the cyber threat landscape for financial institutions," *SWIFT Institute Working Paper*, 2017.

[27] M. R. Hoque, M. S. Rahman, N. J. Nipa, and M. R. Hasan, "Mobile health interventions in developing countries: a systematic review," *Health Informatics Journal*, vol. 26, no. 4, pp. 2792–2810, 2020.

[28] N. T. Msweli and T. Mawela, "Enablers and barriers for mobile commerce and banking services among the elderly in developing countries: a systematic review," in *Responsible Design, Implementation and Use of Information and Communication Technology: 19th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2020, Skukuza, South Africa, April 6–8, 2020, Proceedings, Part II 19*. Springer, 2020, pp. 319–330.

[29] M. Malik, "A review of empirical research on internet & mobile banking in developing countries using utaut model during the period 2015 to april 2020," *Journal of Internet Banking and Commerce*, vol. 25, no. 2, pp. 1–22, 2020.

[30] B. Martínez-Pérez, I. De La Torre-Díez, and M. López-Coronado, "Privacy and security in mobile health apps: a review and recommendations," *Journal of medical systems*, vol. 39, pp. 1–8, 2015.

[31] E. O. Mkpojiogu, A. Hussain, and M. O. Agbudu, "Security issues in the use of mobile educational apps: A review." *International Journal of Interactive Mobile Technologies*, vol. 15, no. 6, 2021.

[32] G. Koala, D. Bassolé, A. Zerbo/Sabané, T. F. Bissyandé, and O. Sié, "Analysis of the impact of permissions on the vulnerability of mobile applications," in *e-Infrastructure and e-Services for Developing Countries*, R. Zitouni, M. Agueh, P. Houngue, and H. Soude, Eds. Cham: Springer International Publishing, 2020, pp. 3–14.

[33] F. Ibrar, H. Saleem, S. Castle, and M. Z. Malik, "A study of static analysis tools to detect vulnerabilities of branchless banking applications in developing countries," ser. ICTD '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3136560.3136595

[34] D. Bassolé, G. Koala, Y. Traoré, and O. Sié, "Vulnerability analysis in mobile banking and payment applications on android in african countries," in *Innovations and Interdisciplinary Solutions for Underserved Areas*, J. P. R. Thorn, A. Gueye, and A. P. Hejnowicz, Eds. Cham: Springer International Publishing, 2020, pp. 164–175.

[35] O. Osho, U. L. Mohammed, N. N. Nimzing, A. A. Uduimoh, and S. Misra, "Forensic analysis of mobile banking apps," in *Computational Science and Its Applications – ICCSA 2019*, S. Misra, O. Gervasi, B. Murgante, E. Stankova, V. Korkhov, C. Torre, A. M. A. Rocha, D. Taniar, B. O. Apduhan, and E. Tarantino, Eds. Cham: Springer International Publishing, 2019, pp. 613–626.

[36] A. Diallo, J. Samhi, T. Bissyandé, and J. Klein. (2024) (in)security of mobile apps in developing countries: A systematic literature review - dataset. [Online]. Available: https://github.com/liounea/Dataset-for-the-SLR-in-mobile-app-security

[37] proofpoint. (2024) What is a data breach? [Online]. Available: https://www.proofpoint.com/us/threat-reference/data-breach

[38] ValueMentor, "Top 10 mobile app security vulnerabilities banks should avoid!" https://valuementor.com/mobile-app-security-testing/top-10-mobile-app-security-vulnerabilities-banks-should-avoid/, accessed: 2025-10-16.

[39] J. Hsu, D. Liu, Y. M. Yu, H. T. Zhao, Z. R. Chen, J. Li, and W. Chen, "The top chinese mobile health apps: a systematic investigation," *Journal of medical Internet research*, vol. 18, no. 8, p. e222, 2016.

[40] S. Latif, R. Rana, J. Qadir, A. Ali, M. A. Imran, and M. S. Younis, "Mobile health in the developing world: Review of literature and lessons from a case study," *IEEE Access*, vol. 5, pp. 11 540–11 556, 2017.

[41] A. Abdullaev, M. A. Al-Absi, A. A. Al-Absi, M. Sain, Y.-S. Lee, and H. J. Lee, "Security challenge and issue of mobile banking in republic of uzbekistan: A state of art survey," in *2019 21st International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2019, pp. 249–255.

[42] N. D. Azeez and M. M. Lakulu, "Review of mobile government at developing countries: benefits and challenges," *International Journal of Economics, Business and Management Research*, vol. 3, no. 2, pp. 198–219, 2019.

[43] M. Rahman, M. Z. Tazim, S. Das, and L. Islam, "State of the art of mobile banking services and future prospects in developing countries," in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2020, pp. 145–149.

[44] R. Pankomera and D. van Greunen, "Opportunities, barriers, and adoption factors of mobile commerce for the informal sector in developing countries in africa: A systematic review," *THE ELECTRONIC JOURNAL OF INFORMATION SYSTEMS IN DEVELOPING COUNTRIES*, vol. 85, no. 5, p. e12096, 2019, e12096 ISD-RA-0111.R3. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/isd2.12096

[45] N. Aljohani and D. Chandran, "The adoption of mobile health applications by patients in developing countries: a systematic review," *International Journal of Advanced Computer Science and Applications*, 2021.

[46] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," Technical report, ver. 2.3 ebse technical report. ebse, Tech. Rep., 2007.

[47] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[48] L. Yang, H. Zhang, H. Shen, X. Huang, X. Zhou, G. Rong, and D. Shao, "Quality assessment in systematic literature reviews: A software engineering perspective," *Information and Software Technology*, vol. 130, p. 106397, 2021.

[49] K. Kant Kamal, P. Joshi, A. Bang, and K. Bhatia, "Effective security testing of mobile applications for building trust in the digital world," in *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*, 2023, pp. 550–556.

[50] T. H. Chiboora, L. Chacha, T. Byagutangaza, and A. Gueye, "Evaluating mobile banking application security posture using the owasp's masvs framework," in *Proceedings of the 6th ACM SIGCAS/SIGCHI Conference on Computing and Sustainable Societies*, 2023, pp. 99–106.

[51] C. W. Munyendo, Y. Acar, and A. J. Aviv, ""desperate times call for desperate measures": User concerns with mobile loan apps in kenya," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 2304–2319.

[52] P. K. Chopdar, "Adoption of covid-19 contact tracing app by extending utaut theory: Perceived disease threat as moderator," *Health Policy and Technology*, vol. 11, no. 3, p. 100651, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211883722000594

[53] G. Vakare, D. Rautela, and H. Shobharam Lamkuche, "User's perception on security and privacy in using crypto currency trading application in india," in *2022 International Conference on Knowledge Engineering and Communication Systems (ICKES)*, 2022, pp. 1–8.

[54] S. S. Bandan, M. Rahman Ajmain, A. R. Rejuan, M. Farhana Khatun, and S. A. Khushbu, "State of survey: Advancement of knowledge environmental sustainability in practicing administrative apps," in *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2022, pp. 1–8.

[55] A. V. Prakash, S. Das, and K. R. Pillai, "Understanding digital contact tracing app continuance: Insights from india," *Health Policy and Technology*, vol. 10, no. 4, p. 100573, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211883721000964

[56] J. R. Kala Kamdjoug, S.-L. Wamba-Taguimdje, S. F. Wamba, and I. B. Kake, "Determining factors and impacts of the intention to adopt mobile banking app in cameroon: Case of sara by afriland first bank," *Journal of*

Retailing and Consumer Services, vol. 61, p. 102509, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0969698921000758

[57] Y. Madwanna, M. Khadse, and B. R. Chandavarkar, "Security issues of unified payments interface and challenges: Case study," in *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, 2021, pp. 150–154.

[58] M. R. Albrecht, J. Blasco, R. B. Jensen, and L. Mareková, "Mesh messaging in large-scale protests: Breaking bridgefy," in *Topics in Cryptology – CT-RSA 2021*, K. G. Paterson, Ed. Cham: Springer International Publishing, 2021, pp. 375–398.

[59] M. N. Al-Ameen, T. Tamanna, S. Nandy, M. A. M. Ahsan, P. Chandra, and S. I. Ahmed, "We don't give a second thought before providing our information: Understanding users' perceptions of information collection by apps in urban bangladesh," in *Proceedings of the 3rd ACM SIGCAS Conference on Computing and Sustainable Societies*, ser. COMPASS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 32–43. [Online]. Available: https://doi.org/10.1145/3378393.3402244

[60] V. Sihag, A. Swami, M. Vardhan, and P. Singh, "Signature based malicious behavior detection in android," in *Computing Science, Communication and Security*, N. Chaubey, S. Parikh, and K. Amin, Eds. Singapore: Springer Singapore, 2020, pp. 251–262.

[61] R. Kumar, S. Kishore, H. Lu, and A. Prakash, "Security analysis of unified payments interface and payment apps in india," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1499–1516. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/kumar

[62] E. D. Ansong and T. Q. Synaepa-Addision, "A comparative study of user data security and privacy in native and cross platform android mobile banking applications," in *2019 International Conference on Cyber Security and Internet of Things (ICSIoT)*, 2019, pp. 5–10.

[63] A. Uduimoh, I. Idris, O. Osho, and S. Abdulhamid, "Forensic analysis of mobile banking applications in nigeria," *i-manager's Journal on Mobile Applications and Technologies*, vol. 6, pp. 9–20, 06 2019.

[64] A. Khatoon and V. Umadevi, "Integrating oauth and aadhaar with e-health care system," in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2018, pp. 1681–1686.

[65] B. Reaves, J. Bowers, N. Scaife, A. Bates, A. Bhartiya, P. Traynor, and K. R. B. Butler, "Mo(bile) money, mo(bile) problems: Analysis of branchless banking applications," *ACM Trans. Priv. Secur.*, vol. 20, no. 3, aug 2017. [Online]. Available: https://doi.org/10.1145/3092368

[66] E.-r. Latifa, E. K. M. Ahemed, and E. G. Mohamed, "Side-effects of permissions requested by mobile banking on android platform: A case study of morocco," in *Proceedings of the 1st International Conference on E-Commerce, E-Business and E-Government*, ser. ICEEG '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 76–81. [Online]. Available: https://doi.org/10.1145/3108421.3108433

[67] F. H. Shezan, S. F. Afroze, and A. Iqbal, "Vulnerability detection in recent android apps: An empirical study," in *2017 International Conference on Networking, Systems and Security (NSysS)*, 2017, pp. 55–63.

[68] S. Kaka, V. N. Sastry, and R. R. Maiti, "On the mitm vulnerability in mobile banking applications for android devices," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2016, pp. 1–6.

[69] A. Khatoon and P. Corcoran, "Privacy concerns on android devices," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2017, pp. 149–152.

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

[71] bertolis. (2022) A step-by-step android penetration testing guide for beginners. [Online]. Available: https://www.hackthebox.com/blog/intro-to-mobile-pentesting

[72] A. Argudo, G. López, and F. Sánchez, "Privacy vulnerability analysis for android applications: A practical approach," in *2017 Fourth International Conference on eDemocracy & eGovernment (ICEDEG)*, 2017, pp. 256–260.

[73] Android. (2023) Permissions on android. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview

[74] ——. (2023) Security with network protocols. [Online]. Available: https://developer.android.com/training/articles/security-ssl

[75] OWASP, "Masvs-code: Code quality," https://mas.owasp.org/MASVS/10-MASVS-CODE/#masvs-code-code-quality, accessed: 2025-10-16.

[76] ——, "Masvs-resilience: Resilience against reverse engineering and tampering," https://mas.owasp.org/MASVS/11-MASVS-RESILIENCE/#masvs-resilience-resilience-against-reverse-engineering-and-tampering, accessed: 2025-10-16.

[77] S. Baur-Yazbeck, J. Frickenstein, and D. Medine, "Cyber security in financial sector development," *CGAP Background Documents*, vol. 5, no. 2, 2019.

[78] A. DIALLO, A. DIOP, A. K. KABORE, A. PILGUN, J. SAMHI, T. BISSYANDE, and J. KLEIN, "On the security of pre-installed android apps in low-cost devices," In press, to appear.

[79] B. Bobe, "Statisticalsales — the malware pre-installed on your phone," https://medium.com/@threatspotlight/episode-5-statisticalsales-417a14e0f75a, accessed: 2025-10-16.

[80] R. Lakshmanan, "Chinese android phones shipped with fake whatsapp, telegram apps targeting crypto users," https://thehackernews.com/2025/04/chinese-android-phones-shipped-with.html, accessed: 2025-10-16.

[81] TIMESOFINDIA, "These android phones come with dangerous pre-installed apps," https://timesofindia.indiatimes.com/gadgets-news/these-android-phones-come-with-dangerous-pre-installed-apps/articleshow/72110567.cms, accessed: 2025-10-16.

[82] C. R. Kurt Knutsson, "Fbi warns over 1 million android devices hijacked by malware," https://www.foxnews.com/tech/fbi-warns-over-1-million-android-devices-hijacked-malware.amp, accessed: 2025-10-16.

[83] M. Zheng, M. Sun, and J. C. Lui, "Droidray: a security evaluation system for customized android firmwares," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 471–482. [Online]. Available: https://doi.org/10.1145/2590296.2590313

[84] BBC, "Chinese phones with built-in malware sold in africa," https://www.bbc.com/news/technology-53903436, accessed: 2025-10-16.

[85] M. Elsabagh, R. Johnson, A. Stavrou, C. Zuo, Q. Zhao, and Z. Lin, "FIRMSCOPE: Automatic uncovering of Privilege-Escalation vulnerabilities in Pre-Installed apps in android firmware," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2379–2396. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/elsabagh

[86] Q. Hou, W. Diao, Y. Wang, X. Liu, S. Liu, L. Ying, S. Guo, Y. Li, M. Nie, and H. Duan, "Large-scale security measurements on the android firmware ecosystem," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 1257–1268.

[87] T. Sutter and B. Tellenbach, "Firmwaredroid: Towards automated static analysis of pre-installed android apps," in *2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, May 2023, pp. 12–22.

[88] J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador, and N. Vallina-Rodriguez, "An analysis of pre-installed android software," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1039–1055.

[89] E. Blázquez, S. Pastrana, A. Feal, J. Gamba, P. Kotzias, N. Vallina-Rodriguez, and J. Tapiador, "Trouble over-the-air: An analysis of fota apps in the android ecosystem," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1606–1622.

[90] Google, “Play Protect Certified Android devices: safe and secure,” 2025. [Online]. Available: httphttps://www.android.com/certified/

[91] Canalys. (2024) African smartphone market surges 24% in q4 2023, despite macro headwinds. [Online]. Available: https://www.canalys.com/newsroom/africa-smartphone-market-Q4-2023

[92] R. Hutchinson. (2024) How to delete pre-installed android apps from your smartphone. [Online]. Available: https://www.geeky-gadgets.com/how-to-delete-pre-installed-android-apps-from-your-smartphone/

[93] Google. (2025) android:exported. [Online]. Available: https://developer.android.com/privacy-and-security/risks/android-exported

[94] U. D. O. LABOR. Guidance on the protection of personally identifiable information (pii). [Online]. Available: https://www.dol.gov/general/ppii

[95] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, “Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps,” in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 259–269. [Online]. Available: https://doi.org/10.1145/2594291.2594299

[96] M. Mitchell, G. Tian, and Z. Wang, “Systematic audit of third-party android phones,” in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 175–186. [Online]. Available: https://doi.org/10.1145/2557547.2557557

[97] S. Sharma. (2025) Android go vs regular android – which is right for your business app? [Online]. Available: https://www.appventurez.com/blog/android-go-apps-vs-regular-apps

[98] Y. Fratantonio, C. Qian, S. P. Chung, and W. Lee, “Cloak and dagger: from two permissions to complete control of the ui feedback loop,” in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 1041–1057.

[99] A. Acar, G. S. Tuncay, E. Luques, H. Oz, A. Aris, and S. Uluagac, “50 shades of support: A device-centric analysis of android security updates,” in *Netw. Distrib. Syst. Security Symp., San Diego, CA, USA*, 2024.

[100] P. Store. (2025) Palm store,let’s play. [Online]. Available: https://www.palmplaystore.com/

[101] A. Documentation, “Apex file format,” https://source.android.com/docs/core/ota/apex, accessed: 2025-12-30.

[102] A. Hu. (2025) From mobile king in africa to e-scooter giant? transsion’s next big move. [Online]. Available: https://www.exportsemi.com/company-post/from-africa-phone-king-to-e-bikes-can-transsion-work-another-miracle/

[103] upstream. (2020) xhelper/triada malware pre-installed on thousands of low cost chinese android devices in emerging markets. [Online]. Available: https://www.upstreamsystems.com/press/press-releases/xhelper-triada-malware-pre-installed-on-thousands-of-low-cost-chinese-android-devices-in-emerging-markets/

[104] T. Debize, "Androwarn: Yet another static code analyzer for malicious android applications," 2012.

[105] P. Lam, E. Bodden, O. Lhoták, and L. Hendren, "The soot framework for java program analysis: a retrospective," in *Cetus Users and Compiler Infastructure Workshop (CETUS 2011)*, vol. 15, no. 35, 2011.

[106] B. Toulas. (2025) Counterfeit android devices found preloaded with triada malware. [Online]. Available: https://www.bleepingcomputer.com/news/security/counterfeit-android-devices-found-preloaded-with-triada-malware/

[107] C. Cimpanu. (2017) Triada trojan found in firmware of low-cost android smartphones. [Online]. Available: https://www.bleepingcomputer.com/news/security/triada-trojan-found-in-firmware-of-low-cost-android-smartphones/

[108] Google. (2025) App components. [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro#components

[109] C. C. W. Enumeration. (2025) Cwe-926: Improper export of android application components. [Online]. Available: https://cwe.mitre.org/data/definitions/926.html

[110] M. Becenti. (2024) Leaky apps: How they're stealing your data without you knowing. [Online]. Available: https://www.quokka.io/blog/leaky-apps-security-risks

[111] Google. (2024) Permission-based access control to exported components. [Online]. Available: https://developer.android.com/privacy-and-security/risks/access-control-to-exported-components

[112] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin, "Permission re-delegation: attacks and defenses," ser. SEC'11. USA: USENIX Association, 2011, p. 22.

[113] H. M. A. Maqsood, K. N. Qureshi, F. Bashir, and N. U. Islam, "Privacy leakage through exploitation of vulnerable inter-app communication on android," in *2019 13th International Conference on Open Source Systems and Technologies (ICOSST)*, 2019, pp. 1–6.

[114] A. Diallo, A. War, M. A. Diouf, J. Samhi, S. Arzt, T. F. Bissyande, and J. Klein, "Security assessment of mobile banking apps in west african economic and monetary union," in *2025 Cybersecurity4D (C4D)*, 2025, pp. 1–13.

[115] A. Turner. (2024) How many people have smartphones in the world? [Online]. Available: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world#

[116] Osiris. (2023) En afrique subsaharienne, le taux d'adoption des smartphones atteindra 87% en 2030. [Online]. Available: http://www.osiris.sn/En-Afrique-subsaharienne-le-taux-d.html

[117] Approov. (2023) Security challenges of financial mobile apps in africa. [Online]. Available: https://approov.io/info/security-challenges-of-financial-mobile-apps-in-africa

[118] K. Pousttchi and M. Schurig, "Assessment of today's mobile banking applications from the view of customer requirements," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the.* IEEE, 2004, pp. 10–pp.

[119] M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, and D. Poshyvanyk, "When and why your code starts to smell bad (and whether the smells go away)," *IEEE Transactions on Software Engineering*, vol. 43, no. 11, pp. 1063–1088, 2017.

[120] A. A. Elkhail and T. Cerny, "On relating code smells to security vulnerabilities," in *2019 IEEE 5th intl conference on big data security on cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE intl conference on intelligent data and security (IDS).* IEEE, 2019, pp. 7–12.

[121] M. Ghafari, P. Gadient, and O. Nierstrasz, "Security smells in android," in *2017 IEEE 17th international working conference on source code analysis and manipulation (SCAM).* IEEE, 2017, pp. 121–130.

[122] J. Bowers, I. N. Sherman, K. R. B. Butler, and P. Traynor, "Characterizing security and privacy practices in emerging digital credit applications," ser. WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 94–107. [Online]. Available: https://doi.org/10.1145/3317549.3319723

[123] S. Bojjagani and V. Sastry, "Vaptai: a threat model for vulnerability assessment and penetration testing of android and ios mobile banking apps," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC).* IEEE, 2017, pp. 77–86.

[124] S. Bojjagani and V. N. Sastry, "Stamba: Security testing for android mobile banking apps," in *Advances in Signal Processing and Intelligent Recognition Systems*, S. M. Thampi, S. Bandyopadhyay, S. Krishnan, K.-C. Li, S. Mosin, and M. Ma, Eds. Cham: Springer International Publishing, 2016, pp. 671–683.

[125] S. Castle, F. Pervaiz, G. Weld, F. Roesner, and R. Anderson, "Let's talk money: Evaluating the security challenges of mobile money in the developing world," in *Proceedings of the 7th Annual Symposium on Computing for Development*, ser. ACM DEV '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/3001913.3001919

[126] B. Reaves, N. Scaife, A. Bates, P. Traynor, and K. R. Butler, "Mo (bile) money, mo (bile) problems: Analysis of branchless banking applications in the

developing world," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 17–32.

[127] BCEAO. (2022, june) Paysage bancaire. [Online]. Available: https://www.bceao.int/fr/content/paysage-bancaire

[128] GlobalStats. (2022) Mobile operating system market share in africa. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/africa/2022

[129] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *Proceedings of the 13th International Conference on Mining Software Repositories*, ser. MSR '16. New York, NY, USA: ACM, 2016, pp. 468–471. [Online]. Available: http://doi.acm.org/10.1145/2901739.2903508

[130] M. Yuen. (2023) Here are the top 50 biggest european banks in 2023. [Online]. Available: https://www.emarketer.com/insights/largest-banks-europe-list/

[131] G. Villaluz and Z. Gull. (2023) 50 largest us banks by total assets, q3 2023. [Online]. Available: https://www.spglobal.com/marketintelligence/en/news-insights/latest-news-headlines/50-largest-us-banks-by-total-assets-q3-2023-79625289

[132] B. Finance. (2023) Banking 500 2023 ranking. [Online]. Available: https://brandirectory.com/rankings/banking/2023/table

[133] A. Bartel, J. Klein, Y. Le Traon, and M. Monperrus, "Dexpler: converting android dalvik bytecode to jimple for static analysis with soot," in *Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program Analysis*, ser. SOAP '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 27–38. [Online]. Available: https://doi.org/10.1145/2259051.2259056

[134] S. Arzt, "Security code smells in apps: Are we getting better?" in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 245–255. [Online]. Available: https://doi.org/10.1145/3540250.3549091

[135] R. Vallee-Rai and L. J. Hendren, "Jimple: Simplifying java bytecode for analyses and transformations," 1998.

[136] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *ACM SIGPLAN Notices*, vol. 49, no. 6. ACM, 2014, pp. 259–269.

[137] J. Samhi, T. F. Bissyandé, and J. Klein, "Androlibzoo: A reliable dataset of libraries based on software dependency analysis," ser. MSR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 32â€“36. [Online]. Available: https://doi.org/10.1145/3643991.3644866

[138] O. M. A. Security. Mobile app cryptography. [Online]. Available: https://mas.owasp.org/MASTG/General/0x04g-Testing-Cryptography/#mobile-app-cryptography

[139] GUARDRAILS. Insecure algorithm. [Online]. Available: https://docs.guardrails.io/docs/vulnerability-classes/insecure-use-of-crypto/insecure-algorithm

[140] T. Mappings, "Cwe-321: Use of hard-coded cryptographic key," *CWE Version 1.11*, vol. 629, p. 420, 2010.

[141] CodeQL. Android webview settings allows access to content links. [Online]. Available: https://codeql.github.com/codeql-query-help/java/java-android-websettings-allow-content-access/#

[142] D. Svoboda. (2014, October) Drd04-j. do not log sensitive information. [Online]. Available: https://wiki.sei.cmu.edu/confluence/display/android/DRD04-J.+Do+not+log+sensitive+information