MOBILE-BASED SOCIAL NETWORKS ENDPOINT SECURITY ENHANCING MODEL

Matoke Nahason Matoke

A Thesis Submitted in Partial Fulfillment of the Requirement of the Degree of Doctor of Philosophy in Information Technology of Masinde Muliro university of Sciences and Technology

November, 2025

DECLARATION

This thesis is my own original work prepared with no other than the indicated sources and support and has not been presentenced elsewhere for a degree or any other award.

Signature:…….. Date:…….

Matoke Nahason Matoke

SIT/H/04/10

CERTIFICATION

The undersigned certify that they have read and hereby recommend for acceptance of Masinde Muliro University of Science and Technology a thesis entitled, "**Mobile-Based Social Networks Endpoint Security Enhancing Model"**

Signature:…….. Date:…….

Dr. Jasper Ondulo, PhD

Department of Computer Science

Masinde Muliro University of Science and Technology

Signature:…….. Date:…….

Dr Daniel Otanga, PhD

Department of Information Technology

Masinde Muliro University of Science and Technology

COPYRIGHT

This thesis is a copyright material protected under the Berne Convention, the copyright Act and under international and national enactments in that behalf, on intellectual property. It may not be reproduced by any means in full or in part except for short extracts in fair dealing so for research or private study, critical scholarly reviews or discourse with acknowledgement, with written permission of the Dean School of Graduate Studies on behalf of both the author and Masinde Muliro University of Science and Technology

DEDICATION

This work is dedicated to my dear loving wife Norah, my daughter Ivy, and my two sons Isaac and Enock.

ACKNOWLEDGEMENTS

I wish to express my profound gratitude to all who supported the successful completion of this work.

My foremost thanks go to the Almighty God, the source of all wisdom and strength, whose divine grace and inspiring words provided the perseverance and guidance needed for this journey. The truth that "with God, all things are possible" was my steadfast anchor and source of encouragement.

I extend my deepest appreciation to my supervisors, Dr. Jasper Ondulo and Dr. Daniel Otanga, for their invaluable mentorship, scholarly insight, and unwavering support. Their intellectual rigor, patient guidance, and dedication of precious time were foundational to this study. Their insightful critiques and advice significantly shaped this work and inspired me throughout the research process.

My heartfelt thanks go to my beloved family for their unwavering belief and unprecedented support, which formed the bedrock of my efforts. Their countless sacrifices, understanding, and constant encouragement provided the motivation and serene environment necessary to achieve my objectives.

Finally, I sincerely thank my friends, colleagues, and the institutional bodies whose contributions, direct or indirect, facilitated this academic pursuit. This work is a testament to the support of a cherished community.

Formatiert: Einzug: Links: 0 cm

ABSTRACT

The pervasive integration of mobile devices and social networking has created a critical security paradigm where the endpoint device has become the primary target for cyber threats. This study addresses the fundamental disconnect between the value of data on mobile devices and the inadequacy of security models focused predominantly on server and network protection. The research was guided by the objective to establish emerging security trends and develop an Endpoint Security Enhancing Model for Mobile-based Social Networks (MbSNs). Employing a pragmatic, mixed-methods approach, the study integrated a quantitative survey of 257 users with qualitative vulnerability simulations of major platforms—WhatsApp, Facebook, and X (formerly Twitter)—using the Mobile Security Framework (MobSF) and GitHub Suite for penetration testing. The investigation yielded several critical findings. The survey revealed a significant "awareness-action gap" among users, who demonstrated knowledge of threats like phishing but exhibited poor security hygiene, with 19.1% using no device password and 84.8% not using a VPN. Crucially, statistical analysis (correlation and linear regression) showed no significant relationship between specific mobile threats and the choice of social networking application (R^2 as low as 1.4%), indicating that vulnerability is universal across platforms. The simulation results provided empirical validation, uncovering high-severity vulnerabilities (CVSS scores 7.0-8.1) rooted not in broken encryption, but in systemic design flaws, including dangerous permission misuse (such as RECORD_AUDIO, CAMERA), buffer overflows, and insecure software components like exported broadcast receivers, which create direct data leakage pathways. In response to these findings, the study designed and implemented the Mobile-based Authentication Technique (MbAT), a novel, layered security model built on a Defense-in-Depth principle. The model anchors its security to a hardware root of trust—the SIM card—and employs a dual-layer encryption strategy. It leverages the robust Signal Protocol for end-to-end encrypted data-in-transit, ensuring forward secrecy and post-compromise security, while utilizing the lightweight Blowfish algorithm for efficient encryption of data-at-rest on the endpoint device. A critical innovation of MbAT is its secure handling of the XML-JSON transformation layer, a necessary interoperability feature, which is compartmentalized and "sandwiched" between robust SIM-based authentication and cryptographic operations to mitigate associated injection and parsing threats. In conclusion, this research successfully re-frames mobile social network security as a socio-technical challenge, demonstrating that the threat landscape is uniform across applications and rooted in the interplay of user behavior, permission models, and architectural flaws. The proposed MbAT model offers a holistic, proof-of-concept solution that transforms the endpoint from the weakest link into a verifiable component of the security architecture. By providing a scalable, user-aware framework that integrates a hardware-anchored root of trust with state-of-the-art cryptographic protocols, this study lays a foundational blueprint for achieving a more secure and privacy-respecting future for mobile social networking.

TABLE OF CONTENTS

TITLE PAGE ..... **i**

DECLARATION ..... **ii**

COPYRIGHT ..... **iii**

DEDICATION ..... **iv**

ABSTRACT ..... **vi**

TABLE OF CONTENTS ..... **vii**

LIST OF TABLES ..... **xii**

LIST OF FIGURES ..... **xv**

LIST OF ABBREVIATIONS AND ACRONYMS ..... **xvi**

DEFINITION OF TERMS ..... **xviii**

CHAPTER ONE:INTRODUCTION **1**

1.1Overview 1

1.2 Background of the Study 1

1.3 Statement of the Problem 7

1.4 General Objective 7

1.5 Research Specific Objectives 8

1.6 Research Questions 8

1.7 Scope of the Study 8

1.8 Significance of the Study 9

1.9 Limitation of the Study 9

1.10 Assumptions 10

1.11. Thesis organization 10

1.12 Study Contributions: 11

CHAPTER TWO:LITERATURE REVIEW **13**

2.1 Overview 13

2.2 Key Concepts 13

2.2.1 Mobile Computing Explained 13

2.2.1.1 Mobile Computing Technologies 14

2.2.1.2 Wireless Networks 17

2.2.1.3 Social Networking 17

2.2.1.4 Mobile Social Network Applications 18

2.2.1.5 Social computing 19

2.2.1.6 Social media 21

2.3 Objectives Analysis based on the literature 23

2.3.1 Mobile Social Networking Vulnerabilities 23

2.3.1.1 Social networking and security risks 24

2.3.1.2 Location cheating and Spoofing Attacks 26

2.3.1.3 Eavesdropping 30

2.3.2 Mobile Social Networking Attack Techniques 35

2.3.2.1 Social engineering 35

2.3.2.2 Direct anonymity 39

2.3.2.3 Indirect anonymity 40

2.4 Theories, Models and Frameworks 42

2.4.1 Mobile social network security techniques 42

2.4.2 Attack Prevention Techniques 43

2.4.3 Security Frameworks in Mobile Social Networking 45

2.4.3.1 WhozThat 48

2.4.3.2 Secure SocialAware Architecture 49

2.4.4 Theoretical Framework 50
2.4.4.1 Security Models 51
2.4.4.1.1 The Signal Protocol 51
2.4.4.1.2 End to End (E2E) 52
2.4.4.1.3 Lightweight Cryptography 53
2.4.4.1.4 Blowfish algorithm 55
2.7 Conceptual Framework 62
2.8 Summary 63

CHAPTER THREE:RESEARCH METHODOLOGY 65

3.1 Overview 65
3.2 Research Design 65
3.3 Research Paradigms: A Methodological Reflection 67
3.4 Research Philosophy: The Pragmatic Paradigm in Practice 68
3.4.1 Pragmatism 69
3.5 Research Approaches 70
3.6 Research Strategy (Study Procedures for the Collection of Quantitative Data) 71
3.6.1 Questionnaire Development Process 72
3.6.2 Data collection method(s) 73
3.7 Research Method 74
3.8 Techniques Used 75
3.8.1 Quasi-Experimental setup [Simulation] 78
3.9 Data Analysis 83
3.10 Validation of the MbAT 84
3.11 Ethical consideration 85

CHAPTER FOUR **87**

DATA ANALYSIS AND INTERPRETATION **87**

4.1 Overview 87

4.2 Findings from the Survey Data (Descriptive Statistics) 87

4.3 Mobile endpoint device, SMN and Operating System in Use 89

4.4 Mode of Security in Use at the EndPoint 91

4.5 Techniques Used by Adversaries to Attack MSN 107

4.6 Challenges in Securing Mobile Social Networks 109

4.7 Quasi-Experimental setup [Simulation] Results and Interpretation. 114

4.7.1 Discussion of Simulation Results 119

CHAPTER FIVE:MOBILE-BASED SOCIAL NETWORKING SECURITY TECHNIQUE **121**

5.1 Overview 121

5.2 Secure Messaging 121

5.2.1 The Signal Protocol 122

5.3 A Model for Mobile-Based Authentication: A Layered Security Approach 124

5.4 Requirements of the MbAT Technique 128

5.4.1 Structure of the inputs 129

5.4.2 Design of the of the MAT Technique 138

5.4.3 Context of the of the MbAT Technique 139

5.5 Architectural Design of the MbAT Technique 139

5.5.1 Input component 140

5.6.2 Activity Diagram 140

5.5.2 Encryption component 141

5.5.3 Authentication component 141

5.5.4 Decryption Component 142
5.5.5 Output Component 142
5.6 Design Model for the MbAT Technique 142
5.6.1 Use Case Design 143
5.6.3 Implemented Algorithms 145
5.6.4 Execution of the for MbAT Technique 148
5.7 Technique Validation Results 150
5.7.1 Architectural Validation: Mobile Network Operator (MNO) Result 151
5.7.2 Functional Validation: University Pilot Report 152
5.7.3 Security Validation: Cybersecurity Red Team Report 154
5.8 Summary 155

CHAPTER SIX:DISCUSSION, CONCLUSION AND RECOMMENDATIONS .. 157
6.1 Discussion 157
6.2 Conclusion 159
6.3 Recommendations 161

REFERENCES 163

APPENDICES 177
Appendix I: Simulation 177
Appendix II: Technique development 234
Appendix III: Questionnaire 238

LIST OF TABLES

Table 2.1: Defense strategies for both developers and end-users 33

Table 2.2: Mobile Security Vulnerabilities, Attacks Vectors, Risk Level and Mitigation Strategies 34

Table 2.3 Comparison of Algorithms 55

Table 3.1 A summary of the methodology and Techniques for each of the four objectives 66

Table 4.1: Distribution of population by Gender 87

Table 4.2: Distribution of population by Age 88

Table 4.3: Distribution of population by Level of Education 88

Table 4.4: Type of mobile endpoint device used 89

Table 4.5: A Summary of Operating System of the endpoint device 90

Table 4.6: Type of Social Networking Sites Regularly Used 91

Table 4.7: Use of Strong Password or Passcode on Endpoint Mobile Device 91

Table 4.8: Use of Multi-Factor Authentication on the Social Networking Accounts 92

Table 4.9: Update of Software or OS 93

Table 4:10: Regular Use of Antivirus or Security Software on the device 93

Table 4.11: Adjusting Privacy Settings after a Breach 94

Table 4.12: Use of VPN on the Endpoint Device 95

Table 4.13: Managing Permission for mobile apps, allowing access to specific features 96

Table 4.14: Awareness of data sharing practices among mobile apps 97

Table 4.15: Awareness of the potential security risks associated with social networking sites 98

Table 4.16: Regularly Checking for Updates and Patches for Mobile Apps 98

Table 4.17: Use of mobile device on unsecured Wi-Fi networks 99

Table 4.18: Having been hacked or having the account compromised while using social networking sites on mobile device. 100

Table 4.19: Receiving suspicious emails, messages, or links on your mobile device while using social networking sites. 100

Table 4.20: Familiarity with privacy settings on mobile device 101

Table 4.21: Crosstabulation on how often one manages permissions for his/her mobile apps such as allowing an app access to a specific feature (like location, camera, or microphone) and Gender 102

Table 4.22: Has experienced any security incidents or data breaches while using social networking sites on mobile device. 103

Table 4.23: Correlation between mobile threats and mobile social network 104

Table 4.24: Linear Regression (Model Summary) 104

Table 4.25: Linear Regression (Model Summary) 105

Table 4.26: Linear Regression (ANOVA) 106

Table 4.27: Linear regression (Coefficients) 107

Table 4.28: Likert scale on techniques used to attack mobile based social network 108

Table 4.29: Challenges in securing MSN 111

Table 4.30: Correlation between mobile attacks technique and mobile social network 113

Table 4.31: WhatsApp Vulnerabilities, Summary 115

Table 4.32: Facebook Vulnerabilities, Summary 116

Table 4.33: X Vulnerabilities, Summary 118

Table 5.1: MbAT's design mitigation for dynamic conversion 136

LIST OF FIGURES

Figure 2.1: Research Gap 57

Figure 2.2 Theoretical Model for MbSN 61

Figure 2.3 Conceptual Framework Author, 2025 63

Figure 3.1a: MobSF v3.4.1- Mobile Security Framework Simulation 77

Figure 3.1b: MobSF v3.4.1- Mobile Security Framework Simulation 77

Figure 3.2: End-to-end encryption layer 81

Figure 4.1: Sample Simulated output for WhatsApp 115

Figure 4.2: Simulated application Details for Facebook 117

Figure 4.3: Sample Permissions for X 119

Figure 5.1: Proposed Model for MbSNs 125

Figure 5.2 Context Diagram for MbAT Technique 139

Figure 5.4 Activity diagram of MbAT Technique 140

Figure 5.3 Use case for MbAT iteration 143

Figure 5.5: Technique execution screen shots showing (a) Factor Authentication, (b) Contacts, (c) Cipher text & (d) Enhanced Security 149

Figure 5.6: The Technique for augmenting Mobile-base Social Networks 150

LIST OF ABBREVIATIONS AND ACRONYMS

3G	Third Generations
API	Application programming interface
ATM	Automated teller machine
BREW	Binary Runtime Environment for Wireless
CDMA	Code Division Multiple Access
E2EE	End to end encryption
GPS	Global positioning system
GSM	Global System of Mobile communication
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
i-mode	Packet based service for mobile phones offered by Japans leader in wireless technology
JISC	Joint Information Systems Committee
PC	Personal computer
PDA	Personal digital assistance
RF	Radio frequency
SMS	Short Message Service

TDMA	Time division multiple access
VPN	Virtual Private Networks
WAP	Wireless Application Protocol

DEFINITION OF TERMS

Encryption: Protects data by making it unreadable without a key.

Endpoints: any devices that connect to a network, including computers, laptops, smartphones, tablets, and even IoT devices

Endpoint security: protecting devices like desktops, laptops, and mobile devices (endpoints) from cyber threats and malicious activity by implementing various security measures and technologies

Enhanced Model: Model that minimizes disruptions caused by security incidents.

CHAPTER ONE

INTRODUCTION

1.1Overview

This chapter chronologically discusses and presents the background of the study, statement of the problem, objectives, research questions, scope of the study, significance of the study, limitations of the study, assumptions and thesis organization.

1.2 Background of the Study

The rapidly expanding market of mobile devices and their open programming platforms offer corporations significant opportunities to interact with clients and customers (Anderson & Rivera, 2018). These devices', rich functionality, supports creative innovations that are not possible through a traditional PC application. However, size and computing power limitations have forced companies to redesign their Internet presence to provide mobile device users a browsing experience comparable to that of the PC (Lee & Kim, 2019). As developers redesign websites and create mobile applications, they need to consider the potential security risks and mitigate them (Anderson & Rivera, 2018; Chen, 2020). This is particularly critical given the significant increase in the popularity and ubiquity of mobile devices among users all around the globe (Global Mobile Insights, 2021).

These devices are based on a specific operating system, which enables users to install a vast variety of applications, commonly referred to as "apps," from online sources called markets like the Apple App Store and Google Play (Zhang, 2017). These apps are the essence of smartphones, enriching their functionality and enhancing the everyday lives of their users. The app markets allow users to perform a quick search and installation of new apps, but at

the same time, they are also a source of different kinds of malware disguised as normal apps (Parker & Johns, 2022). Nowadays, mobile devices are subject to a wide range of security challenges and malicious threats (Zhao, 2020).

A mobile endpoint security solution is solely concerned with the security of mobile devices, including smartphones and tablets. They often hold sensitive information and connect with corporate networks, becoming one of the prime targets of cyber threats. A recent survey points to a growing concern amongst cybersecurity experts, where 97% of them believe that malicious mobile apps could potentially result in sensitive data leaks (Cybersecurity Ventures, 2023).

Endpoint security protection began life in the 1980s, when attacks like the Vienna virus were first launched against computers. These early viruses focused mainly on self-reproduction and seldom contained malicious code designed to destroy systems or steal information (Harley, 2018).

Two significant but independent trends in recent years are the popularity of social networking applications and the adoption of mobile devices. Mobile social networking (MSN) is social networking where individuals with similar interests converse and connect with one another through their mobile phone and/or tablet. The current generations of phones are pocket computers when compared to their predecessors (Thompson, 2019). They are relatively well-resourced, and modern mobile phones are small computers not just in size but also in functionality. In addition to voice and simple text message communication, they can include a multitude of other features, such as using web services, sending and receiving e-mails with attachments, and playing multi-player online games (Roberts, 2021).

Social network sites are web-based services that allow individuals to construct a public or semi-public profile within a bounded system, articulate a list of other users with whom they share a connection, and view and traverse their list of connections (Ellison & Boyd, 2013). The nature and nomenclature of these connections may vary from site to site (Ellison & Boyd, 2013). It is important to note that social media has grown rapidly overtime; today, a vast majority of active Internet users visit social networks and blogs (Nielsen, 2022). Social Networks have undergone dramatic growth in recent years, with a large percentage of social media users accessing content from their mobile phones (Smith & Galloway, 2021). Social networks have had an almost viral growth of users during the last few years, notably Facebook, Twitter, LinkedIn, and WhatsApp, enabling users to share personal information with family members, friends, and colleagues as well as loose acquaintances and the public in general (Davis, 2020). Therefore, social networks provide an extremely suitable space to instantly share multimedia information between individuals and their neighbors in the social graph, providing a powerful reflection of the structure and dynamics of the society of the 21st century (Miller et al., 2019).

The personal information most commonly consists of text notes/messages, photos and videos, but it can also consist of social data such as contacts or list of friends (Wilson, 2018).

It is notable that Social Networking Internet services are changing the way people communicate with one another, entertain and live. They are a primary reason that many people have become avid Internet users (Green et al., 2021). The rapid growth of mobile computing has been made possible through advances in mobile terminals connected through wireless networks, which has enabled users to be ubiquitously connected and thus be ‘always on’ (Kumar & Patel, 2019).

The rapid development in mobile computing has made social networks almost ubiquitous, where access to them can be done through multiple devices such as a personal computer (PC) or a mobile phone (Lee, 2020). While it is possible to access social networking services from almost anywhere, few services can today take full advantage of the mobility of the users (Ibrahim, 2022).

Except for the well-known “first tier” social networks with hundreds of millions of users that span the entire world, there are also many smaller social networking sites that are equally as popular within a more limited geographical scope, for example, within a city, country or continent (Johansson, 2018).

Security is a system attribute that reflects the ability of the system to protect itself from external attacks, which may be accidental or deliberate (Pfleeger & Pfleeger, 2018). These external attacks are possible because mobile devices are networked and are therefore accessible everywhere; that is to say, they are pervasive or ubiquitous (White, 2020). Virtually any element of cyberspace can be at risk, and the degree of interconnection of those elements can make it difficult to determine the extent of the security that is needed (National Research Council, 2019). Identifying the major weaknesses in security is an area of some controversy. However, some components appear to be sources of potential significant risk because either major vulnerabilities have been identified or substantial impacts could result from a successful attack (Anderson, 2001).

Today’s advanced mobile devices are well integrated with the Internet and have far more functionality than mobile devices of the past. They are increasingly used in the same way as personal computers (PCs), potentially making them susceptible to similar threats affecting PCs connected to the Internet (Yang, 2021). Since mobile devices can contain vast amounts

of sensitive and personal information, they are attractive targets that provide unique opportunities for criminals' intent on exploiting them (Fowler, 2022). Both individuals and society as a whole can suffer serious consequences if these devices are compromised.

People may use social networking services for different reasons: to network with new contacts, reconnect with old friends, maintain current relationships, build or promote a business or project, participate in discussions about a certain topic, or just have fun meeting and interacting with other users (Barnes, 2019). Some services, such as Facebook and Twitter, have a broad range of users, while others cater to specific interests (Facebook, 2023).

Social Media at Work Place

New social networking and social media technologies are widely believed to offer business and governmental organizations a powerful means to improve their communications, processes and, ultimately, performance (McKinsey & Company, 2021). Popular business and technology journals are especially optimistic about the positive impacts that these technologies will have on the organizations that adopt them. But these potential benefits are not well understood, partly due to the lack of empirical research on organizations' use of the technologies and platforms that underlie social networking and social media (Kane et al., 2022).

In the current digital and social media world, the use of social media, blogging, video tablets, and smartphones is only growing in popularity. This new level of personal and professional transparency combined with the availability of mobile technology heightens the likelihood that the workplace will be impacted with social media in one way or another (Thompson,

2020). It is likely that the recruiting team is using social media as a candidate recruiting and sourcing technique (HR.com, 2023). This shows that organizations are using internal and social media collaborative communication techniques to increase productivity and connectivity. However, this integration brings significant privacy and security challenges for both employees and employers (Wright, 2021). Due to the increase of employee firings, layoff announcements, and terminations that are happening through social media channels, it is clear that social media use in the workplace must be carefully managed (SHRM, 2022)

The Internet connects the whole world over a digital network, which makes it more difficult to protect information using traditional technical solutions (Zheng, 2019). Although a number of mobile-based security frameworks have been proposed, these are often dependent on specific networks, which mean that they cannot solve a problem outside their intended networks (Patel & Li, 2020). There is a need for a holistic, network-independent framework to solve social mobile network security problems (Garcia, 2021).

Based on the analysis, all current security frameworks possess critical, disqualifying flaws. OS-native models are vulnerable to social engineering as their application sandboxing cannot stop users from being tricked. Enterprise and advanced Mobile Threat Defense solutions are either network-dependent or create privacy trade-offs, and neither can fully mitigate sophisticated social scams on encrypted platforms. Therefore, no single framework achieves complete, network-independent protection across all threat vectors without significant compromise. Therefore, this study proposes a mobile-based social networking security framework and threat prevention techniques that will be network-independent.

1.3 Statement of the Problem

The central problem in mobile-based social network security is a critical disconnect: mobile devices are the primary repository for valuable data, but security models remain disproportionately focused on server-side and network-level protections, leaving the mobile endpoint itself as a vulnerable and overlooked attack vector (IBM Security, 2023). This vulnerability is exploited through a combination of technical and human factors. Globally, endpoints are the frontline of cyber conflict, with phishing involved in 41% of incidents and the human element playing a role in 74% of all breaches (IBM Security, 2023; Verizon, 2023). The scale of the threat is significant, with one in every 35 organizations facing a weekly mobile malware attack (Check Point Software Technologies, 2023). The issue is acute in mobile-first economies like Kenya, which suffered KES 5.2 billion in losses in 2022 and over 1.2 billion detected cyber threats in a single quarter (Serianu, 2022; Communications Authority of Kenya, 2023). The threat is further impacted by a 3.5-fold global increase in mobile banking Trojans (Kaspersky, 2023). Consequently, the "social mobile network security problem" is defined by the convergence of exploitable human psychology and sophisticated, targeted malware, making the endpoint the primary battleground for data compromise. Therefore, the study sought to address the critical security gap where mobile devices, as primary data repositories, are overlooked as vulnerable endpoints, and to investigate the mitigation and the impact of security on these devices, addressing the current oversight of endpoint security.

1.4 General Objective

The general objective of this study was to establish the emerging security trends in mobile-based social networks with a view to developing an Endpoint Security Enhancing Model.

1.5 Research Specific Objectives

i. To establish the vulnerabilities in today's mobile-based social networks

ii. To determine techniques used by adversaries to attack mobile-based social networks

iii. To establish mechanisms used to mitigate the vulnerabilities in mobile-based social networks at endpoint device

iv. To develop a model to augment a mobile-based social networking security mechanism at the endpoint device

1.6 Research Questions

i. What are the vulnerabilities, in today's mobile-based social networks?

ii. What are the attack techniques used by adversaries on mobile-based social networks?

iii. What are the mechanisms used to mitigate vulnerabilities in mobile-based social networks?

iv. How can a model to augment security mechanism on mobile-based social network be developed?

1.7 Scope of the Study

This study was scoped to develop and validate an endpoint security enhancing model for mobile-based social networks. The research specifically focused on the mobile device as the primary unit of analysis, investigating vulnerabilities within mobile operating systems (Android and iOS) that are exploited to compromise social network data. This included analyzing insecure data storage, insufficient encryption, and weaknesses in the app ecosystem pertinent to applications like WhatsApp, Facebook, and X (Twitter).

The study employed a three-phase methodology to address this endpoint-specific problem. A survey design was first used to diagnose security risks and user behaviors within the target population. The findings from this survey then informed a quasi-experimental design to identify critical security gaps. Finally, these empirical results directly guided a design science approach to create, develop, and rigorously validate the novel endpoint security enhancing model.

1.8 Significance of the Study

The need for Mobile Endpoint Security has become critical because mobile devices have become essential for both personal and professional purposes. The center of communication now lies with these devices regarding mailing, scheduling, and handling sensitive information.

The security research on mobile-based social networking could not have been carried out at a more appropriate time. The research is timely because of the concern for mobile social networking has never been so central to online transactions. The use of mobile-base social networks is becoming a key aspect of dissemination and sharing of knowledge to diverse communities, the more secure, the more confidence, the mobile-based social networks will give to the use of the platforms. Therefore, it is the right time for the study, as the findings addresses the gap in the mobile-based social networking problems in developing countries. The study supports in streamlining policy making and bring about best practices in a better communication environment.

1.9 Limitation of the Study

The study acknowledges certain inherent limitations that shaped its findings. Its exclusive focus on Android and iOS platforms, along with specific applications, necessarily restricted the direct generalizability of its results to other systems; this was however addressed by clearly delineating the research's scope and articulating the methodology in a platform-agnostic manner to aid future

replication. Further, the initial reliance on survey data carried the risk of user bias, a concern mitigated by employing validated instruments and triangulating self-reported data with objective experimental findings. Finally, recognizing that the dynamic cyber threat landscape could date specific findings, the study prioritized the development of an adaptable, principle-based model to ensure its long-term relevance and robustness.

1.10 Assumptions

The researcher assumed that users are generally unaware of the full extent of privacy and security risks associated with these platforms. Additionally, assumed that users are more vulnerable to social engineering attacks due to the increased reliance on mobile devices and the potential for less rigorous security practices compared to desktop environments. Further, the study assumed that the distributed nature of mobile networks and the variety of devices used create unique security challenges compared to traditional, centralized networks.

1.11. Thesis organization

The thesis has six chapters. It begins with the preliminary section including the declaration, copyright, declaration, Acknowledgement, abstract, the table of contents, list of figures and the list of abbreviation.

This is followed by chapter one which has the introduction. This includes the background to the study, statement of the problem, general objective, research question, scope of the study, assumptions, limitations and the structure of the thesis.

Chapter two has the literature review on mobile-based social networks and mobile social networking, vulnerabilities that can be exploited on the mobile-based social networks,

techniques use by adversaries to exploit or the mobile-based social networks, various existing frameworks of social networks and theoretical framework.

Chapter three is the research methodology. This includes the location of study, study population, research design and sampling strategy among others.

Chapter four the data analysis and result presentations

Chapter five the proposed Mobile based social networks security techniques.

Chapter six presents the discussion, conclusion and recommendations.

1.12 Study Contributions:

The researcher contribution can be viewed in the context of providing valuable insights into the unique security challenges posed by mobile social networks, leading to the creation of a more robust and tailored endpoint security solution specifically designed for this environment, potentially including the following contributions:

1. Developed a model to augment a mobile-based social networking security mechanism at the endpoint device.
2. Identification of New Threats: The researcher analyzed current trends in mobile social network usage and vulnerabilities; thus, the study was able to identify emerging threats that are specifically targeted at mobile devices or social networking platforms, allowing for proactive security measures to be developed against them.
3. Understanding User Behavior: The study investigated how user behavior on mobile social networks contributes to security risks, such as oversharing personal information or frequent

use of insecure third-party applications, allowing for the development of security solutions that address these behaviors.

4. Developing Targeted Security Measures: Based on the identified threats and user behaviors, the study could propose specific enhancements to existing endpoint security models, including advanced threat detection mechanisms, improved data encryption, and secure authentication procedures tailored to mobile social networking environments.
5. A Holistic Approach to Security: By considering both the network and user perspectives, the study was able to give a more holistic approach to endpoint security on mobile devices, ensuring that not only the device itself but also the data and user interactions are protected.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

This chapter presents a structured security analysis of mobile social networks, first defining the mobile device as the primary vulnerability. It subsequently details a landscape of threats like location spoofing and social engineering, while reviewing existing cryptographic and architectural countermeasures. The analysis ultimately identifies a crucial gap: the absence of a model integrating technical solutions with human and platform-dependent factors. Consequently, it proposes a new theoretical and conceptual framework to guide the creation of a holistic security solution, thereby establishing the foundation for its contribution.

2.2 Key Concepts

The key concepts collectively describe the transformative synergy between mobile connectivity and social interaction. Mobile computing, powered by its technologies and wireless networks, provides the ubiquitous infrastructure that allows social computing to move beyond the desktop and into everyday life. This fusion is realized through mobile social network applications, a dominant form of social media, which have fundamentally reshaped communication, information dissemination, and community formation. The scientific importance lies in this ecosystem's role as a massive, real-time laboratory for studying human behavior, network dynamics, and the development of next-generation context-aware technologies.

2.2.1 Mobile Computing Explained

Unlike mobile devices in the past, which were designed for the sole purpose of voice-based communication, today’s phones are powerful devices that can communicate, compute, and

sense. The sensing capabilities of mobile phones come from a wide array of audio, video, location, and motion sensors, including microphones, multiple high-resolution cameras, GPS, and accelerometers. While these sensors enable a variety of new and immersive applications, they also present a significant and expanding attack surface that can seriously jeopardize user privacy (He et al., 2023).

The core threat remains that if a mobile device is compromised, an adversary can not only exfiltrate the sensitive and enormous data stored on the device but also perpetually monitor the user's environment and actions by stealthily sniffing the sensors (Zhang et al., 2022). Modern malware can leverage these sensors for side-channel attacks; for instance, using the accelerometer to infer keystrokes or the gyroscope to eavesdrop on conversations (Chen et al., 2021). This persistent sensing capability transforms the smartphone from a mere data repository into a powerful surveillance tool in the hands of an attacker (Kumar et al., 2019). With the increasing volume of sensitive data—from financial information to biometrics—stored on these devices, their vulnerability is more critical than ever, a trend confirmed by the consistent rise in mobile malware focused on data theft (Sophos, 2024). The evolution not only brought convenience but also introduce new vectors for exploration, which the following sections explore in greater depth.

2.2.1.1 Mobile Computing Technologies

There are several technologies that have a similar background, which have contributed to the growth of social mobile networks, among them include:

Smartphones, which now allow billions of people to be connected to the Internet continuously and support mature development environments for third-party application

developers. This has put significant personal computing power in the pockets of users and, at the same time, given them ubiquitous access to rich online social network information. In many urban and increasingly rural areas, there are now high concentrations of active social network users with smartphones, creating a dense fabric of connectivity (Pew Research Center, 2021).

The dramatic rise in smartphone usage has continued unabated, with these devices becoming the primary gateway to the internet for a majority of the global population. This dominance is no longer led by a single platform but by a robust ecosystem of Android and iOS devices. As of 2023, mobile devices (excluding tablets) generated 58.33% of all global website traffic, solidifying their position as the main tool for digital interaction (Statista, 2024). The proliferation of these powerful, sensor-rich devices has fundamentally expanded the attack surface for cyber threats (ENISA, 2023).

It is widely recognized that mobile computing devices contain the same vulnerabilities as traditional computers, but they also introduce unique risks. These include the use of cellular and Wi-Fi airwaves instead of wired connections, which are susceptible to interception; denial-of-service attacks tailored for wireless protocols; and the hosting of sensitive financial and identity applications (e.g., digital wallets, mobile banking) that are particularly attractive to hackers (Zhang et al., 2022). The convergence of these factors creates a complex threat landscape for mobile endpoints.

Mobile devices are ubiquitous; global smartphone penetration is expected to exceed 80% by 2025 (Ericsson, 2023). Their usage continues to grow as the technology becomes increasingly powerful. Modern smartphones integrate the functionality of disparate

technologies such as high-quality cameras, advanced gaming systems, and health monitors, all while seeing consistent improvements in storage capacity, processing power, and battery life.

The app store model, pioneered by Apple and Google, has matured into the dominant distribution channel for software. These stores provide a centralized location to download millions of applications that extend device functionality far beyond its out-of-the-box capabilities. While mobile devices have long been able to run third-party apps, the sophistication of these applications has grown exponentially. Modern mobile apps now leverage technologies like augmented reality, machine learning, and real-time collaboration, and are deeply integrated into corporate IT environments for accessing internal systems (Gartner, 2023).

Utilizing third-party applications from app stores, smartphones are increasingly capable of providing much of the same core functionality as a conventional laptop, with additional unique advantages. Mobile devices are almost always with the user, always turned on, and (in most areas) always connected to a data network. This "anytime, anywhere" access is causing an increasing number of people to rely on their mobile device as their primary computer while traveling. While this provides immense productivity benefits through instant access to corporate email, files, and business applications, it simultaneously presents profound security risks, including data leakage, unauthorized access, and malware infection from poorly vetted apps (OWASP, 2021; Lookout, 2024).

2.2.1.2 Wireless Networks

Wireless networks have evolved significantly but are primarily built upon several key technology families: cellular networks (from 4G LTE to 5G NR), wireless local area networks (WLANs) using the 802.11ax (Wi-Fi 6) and newer standards, and personal area networks like Bluetooth. While these technologies have advanced in speed and latency, they continue to share a fundamental security challenge: they rely on the open medium of radio frequency (RF) for transmission. This inherent characteristic makes them susceptible to eavesdropping, jamming, and various signal manipulation attacks, potentially compromising the confidentiality and integrity of data transmissions (Khan et al., 2022; Li et al., 2023).

The modern wireless ecosystem is built upon globally standardized but complex technologies. Key standards include the 3GPP-defined 5G New Radio (NR) for cellular communications and the IEEE 802.11 family (e.g., Wi-Fi 6 and Wi-Fi 7) for local area networking. Each generation introduces new security features, such as enhanced subscriber privacy in 5G, but also unveils novel attack vectors, requiring continuous analysis of their security architectures and the development of robust intrusion detection systems (Arfaoui et al., 2021; Kumar & Singh, 2024).

2.2.1.3 Social Networking

Tim Berners-Lee's original vision of the Web as a medium for collective intelligence has been realized and amplified through the rise of social networks and cloud computing. This growth is driven by a synergy of societal shifts towards digital interaction and technological advancements that enable user co-creation of content and services (Dwivedi et al., 2021). The cloud computing paradigm is the foundational infrastructure that powers this collective creation, scaling to support billions of user-generated interactions.

This is particularly evident in social networking applications, where users actively contribute personal data and, in some cases, participate in the development of services through open APIs and platforms. The landscape has evolved from early examples like Flickr and Digg to dominant, algorithmically-driven platforms like TikTok, Instagram, and X (formerly Twitter), which harness network effects and user-generated content on an unprecedented scale (van Dijck, 2021). These platforms function as modern "mediated publics"—digital environments where people can gather, socialize, and perform identity, all structured by the platform's technical and commercial architecture (boyd, 2010, as cited in later analyses).

The growth of social networks has not only exploded in user count but also in their societal impact, influencing politics, culture, and commerce. The current evolution points towards more decentralized and federated models (such as activity pub protocols) and the integration of immersive technologies, raising new questions about data sovereignty, algorithmic bias, and digital well-being (Tomašev et al., 2020; Pentina & Tarafdar, 2024).

2.2.1.4 Mobile Social Network Applications

The integration of social network information with user location and context has evolved from simple mobile access to sophisticated, context-aware systems. Early approaches primarily focused on extending desktop social network functionality to mobile devices through native applications, which were essentially optimized interfaces for mobile access (Beach et al., 2009).

Modern applications, however, have moved beyond this paradigm. The current integration is deep and pervasive, characterized by several key developments. First, social media platforms are no longer merely accessible on mobile devices; they are

fundamentally *designed* for mobile-first consumption, with features like Stories, ephemeral content, and vertical video formats that prioritize smartphone use (Bayer et al., 2020). Second, the concept of context-awareness has matured. Applications now leverage a sensor fusion of GPS, Bluetooth beacons, and device usage patterns to deliver highly personalized content and location-based features, such as hyper-local news feeds or event recommendations (Sheng et al., 2023). This represents a shift from simply providing access to embedding social interactions within the user's immediate physical and social context.

Finally, the integration has become bidirectional and OS-deep. Social features are now embedded operating-system-wide through shared APIs, allowing for single-sign-on, seamless content sharing across apps, and social graph-based functionalities that extend beyond a single platform. The rise of platforms like TikTok and BeReal exemplifies this, where the core value proposition is a mobile-native social experience driven by algorithmic curation of user-generated content, heavily reliant on real-time context and location (Vaterlaus et al., 2022; Zhang & Liu, 2024). Thus, the modern mobile-social integration is a complex ecosystem of always-on, sensor-driven, and algorithmically-mediated interactions.

2.2.1.5 Social computing

In the age of Web3.0, internet users are leveraging a new suite of decentralized technologies to create and sustain virtual communities and economies built around shared interests and ownership. Communications channels have evolved from blogs and forums to decentralized social protocols and token-gated communities, amplified by blockchain-based mechanisms for credentialing and value exchange. This new paradigm of collective intelligence, often referred to as decentralized autonomous organizations (DAOs) or network states, leverages

cryptographic proofs and tokenized incentives to coordinate human activity, aiming to create systems where the collective output can exceed the sum of its parts in a verifiable and self-sovereign manner (Wang et al., 2021). The potential of this model extends beyond content to the collective provisioning of public goods and the formation of digital nations (Buterin, 2021; Vitalik 2022).

The growth of this modern social computing landscape is driven by the maturation of decentralized infrastructure, including blockchain scalability solutions and zero-knowledge proofs, rather than just broadband and powerful PCs. Collectively, these technologies represent a potential new phase in the evolution of the web, with profound implications for social and business models. While early business interest in Web3 has centered on cryptocurrency and NFTs, the potential impact extends far beyond to areas like decentralized identity, creator economies with improved monetization, and new organizational structures (Chen et al., 2022).

A wide range of decentralized applications (dApps) and Web3 protocols have emerged, introducing new models for interaction and communication. These technologies enable large-scale, trust-minimized collaboration, user-owned data ecosystems, and the programmatic creation of digital communities, thereby transforming how organizations and social networks can be constituted (Fenwick & McCahery, 2023).

Contemporary research on social computing now reflects the shift to decentralized paradigms. The focus has bifurcated: one stream concentrates on the technological issues, user interfaces, and adoption challenges around decentralized identity, key management, and the user experience of self-sovereignty (Zhang et al., 2023). The other stream leverages

advanced computational techniques, particularly agent-based modeling and network science, to simulate and study the emergent behaviors and tokenomic equilibria within these new digital societies (Epstein, 2023).

Modern social computing, particularly in its Web3 incarnation, shifts computation and data ownership to the edges of the network. It aims to empower individual users by enabling them to truly own their digital assets and social graphs, manifest their creativity in new metaverse and NFT contexts, engage in global coordination through DAOs, contribute expertise for tokenized rewards, and assimilate collective bargaining power through governance tokens, all while relying on cryptographic verification rather than trusted intermediaries (Benkler, 2022).

While platforms like Facebook and Flickr still dominate, the landscape is being challenged by a new generation of decentralized social platforms and protocols. Brand names like Farcaster, Lens Protocol, and decentralized versions of existing platforms are extending social interaction beyond the walled gardens of traditional apps and into a more open, composable social graph that is native to the blockchain ecosystem (Zhang & Liu, 2024).

2.2.1.6 Social media

This refers to the use of web-based and mobile technologies to turn communication into an interactive dialogue. Andreas Kaplan and Michael Haenlein have updated their perspective, defining social media as "a group of Internet-based applications that build on the ideological and technological foundations of Web 3.0, and that allow the creation and exchange of User Generated Content" (Kaplan & Haenlein, 2020, p. 2). Social media are media for social interaction, as a superset beyond social communication. Enabled by ubiquitously accessible

and scalable communication techniques, social media substantially change the way of communication between organizations, communities, as well as individuals, a transformation accelerated by the integration of AI and immersive technologies (Dwivedi et al., 2023).

Social media has exploded as a category of online discourse where people create content, share it, and network at a prodigious rate. While platforms like Facebook and Twitter remain prominent, the landscape now includes algorithm-driven platforms like TikTok and decentralized networks. Because of its ease of use, speed, and reach, social media continues to set trends and agendas in society, influencing everything from politics to entertainment, though now through increasingly personalized and AI-curated feeds (Peters, 2023).

Mobile social networking is social networking where individuals with similar interests connect and communicate primarily using a mobile device. Much like web-based social networking, mobile social networking occurs in virtual communities. The industry has evolved from the two-tiered "on-deck" and "off-deck" model to a more complex ecosystem dominated by native apps distributed through the iOS App Store and Google Play Store, which act as the new gatekeepers (Statista, 2024).

Advances in software and hardware technologies have profoundly facilitated the existence of these mobile virtual communities. Modern enabling technologies include high-speed 5G connectivity, sophisticated smartphone operating systems (iOS and Android), cloud computing, and AI-powered recommendation engines that personalize the social experience (Andrews et al., 2024).

Based these key concepts of computing, computing fundamentally shapes mobile user behavior by transforming smartphones into powerful, always-on, sensor-rich devices that provide "anytime, anywhere" access to the internet and sophisticated applications, fostering a reliance on them as primary tools for communication, social networking, and immersive experiences; however, this very computing power creates severe security risks by expanding the attack surface, where the proliferation of sensors and wireless connectivity can turn a compromised device into a surveillance tool, enabling adversaries to exfiltrate sensitive data and perpetually monitor users through side-channel attacks and malware.

2.3 Objectives Analysis based on the literature

This is a systematic research process for defining a study's aims. It involves a comprehensive review of existing scholarly work to identify well-established knowledge, persistent gaps, and unresolved questions. The analysis ensures that research objectives were precisely formulated to address the specific gaps, thereby contributing and adding value to the research. Consequently, the derived objectives are empirically grounded, theoretically justified, and designed to advance collective scientific understanding

2.3.1 Mobile Social Networking Vulnerabilities

The functional convergence between mobile devices and traditional PCs is now complete, with smartphones serving as primary computing platforms for a majority of users. Consequently, the threat landscape has fully merged; mobile devices are equally, if not more, susceptible to a wide range of cyber threats, including sophisticated malware, phishing, and network-based attacks that were once primarily associated with desktop systems (Wei et al., 2022). The risk is amplified because the aggregation of sensitive information on a mobile device is both broader and more enticing. Users routinely leverage mobile portability for

high-value activities like mobile banking, secure corporate email access, and managing digital identities. Furthermore, mobile devices generate unique and highly personal data streams not typically available from a static PC, including continuous location information (GPS), comprehensive communication logs (calls, texts, messaging apps), and biometric data, making them a rich target for attackers (Zhang et al., 2022). The list of threats continues to evolve rapidly, with a notable increase in mobile banking trojans, subscription fraud malware, and espionage-focused threats that exploit these device-specific features (Lookout, 2024).

Despite technological advancements, the challenge of programmatically collecting social data persists due to the entrenched lack of standardization and restrictive data access policies. While initiatives like the Open Social API and ActivityPub (which underpins the Fediverse) represent progress toward interoperability, they have not achieved universal adoption among major platforms (Zhang & Liu, 2024). In practice, access to social data remains heavily constrained by the "walled gardens" of major platform owners, who control APIs through rate limiting, changing data schemas, and monetization strategies that restrict free academic or third-party access (Freelon, 2023). This has created a significant research and development bottleneck, shifting efforts towards decentralized social networking models and privacy-preserving computation techniques that can operate within these constraints (Pentina & Tarafdar, 2024).

2.3.1.1 Social networking and security risks

The prevalence and utility of social networking app have become a foundational element of modern digital life. The professional utility of platforms like LinkedIn for career development, the role of X (formerly Twitter) in public discourse and marketing, and the

social connectivity offered by Facebook and others are well-established (Pew Research Center, 2024). These platforms are leveraged for professional networking, revenue generation, public safety communication, and personal relationship maintenance. However, the integration of these services into daily life necessitates a critical and continuous assessment of their security and privacy implications (Zhang & Liu, 2024). Each platform introduces a unique set of risks that can compromise both personal data and organizational information systems. This research examines these evolving risks and identifies potential mitigation strategies to safeguard data.

The contemporary SN apps landscape has diversified beyond the original triad. While the platforms mentioned remain significant, the ecosystem now includes visually-oriented platforms like Instagram and TikTok, which have seen massive growth and present distinct security challenges, particularly related to data collection and influencer fraud (Peters, 2023). The traditional categorization of platforms has also blurred; Facebook and X are used for both personal and professional purposes, while newer platforms defy simple classification. The fundamental mechanics of forming connections—whether through "Friends," "Followers," or "Connections"—continue to create expansive social graphs that are prime targets for data harvesting, social engineering, and the spread of misinformation (Freelon, 2023).

A critical distinction exists between the risks social networking apps pose to individuals versus corporate systems. For individuals, the primary threats are personal and privacy-centric, manifesting as identity theft, financial fraud, doxxing (the malicious publication of private information), and targeted harassment, which exploit the vast amounts of personal data shared within expansive social graphs (Pew Research Center, 2024; Zhang & Liu,

2024). In contrast, the threats to corporations are systemic and operational, where the same social platforms serve as a vector for large-scale data breaches, intellectual property theft, and corporate espionage facilitated by social engineering attacks like whaling and phishing that compromise entire enterprise systems (Verizon, 2023; ENISA, 2023). Furthermore, corporations face significant reputational damage and financial loss from coordinated misinformation campaigns or the spread of malware through compromised corporate accounts, representing a threat to the organization's infrastructure and public standing rather than the privacy of a single person (Schoen et al., 2023; Proofpoint, 2024).

Social networking applications have become a foundational element of modern digital life, with their utility for professional networking, marketing, and social connectivity being well-established, yet their integration necessitates a critical assessment of security implications. The contemporary landscape has diversified and blurred traditional categories, with the fundamental mechanics of connection-building creating expansive social graphs that are prime targets for data harvesting, social engineering, and misinformation. A critical distinction emerges in how these risks manifest: for individuals, threats are primarily personal and privacy-centric, including identity theft, doxxing, and harassment that exploit shared personal data; whereas for corporations, the threats are systemic and operational, serving as vectors for large-scale data breaches, intellectual property theft, corporate espionage, and reputational damage from coordinated misinformation campaigns or malware spread through compromised accounts.

2.3.1.2 Location cheating and Spoofing Attacks

The fundamental vulnerability to location spoofing stems from the inherent trust that applications place in client-side location data without robust server-side verification

mechanisms. Attackers can exploit the open nature of mobile operating systems like Android to manipulate location services. This is achieved through various methods, including the use of mock location applications, the modification of GPS-related APIs, or the exploitation of debugging interfaces to inject falsified coordinates (Zhang et al., 2022). Even when applications deploy basic client-side anti-cheating checks, these defenses are often insufficient. The rules can be bypassed by sophisticated tools or by running the application on a rooted or jailbroken device, which grants the user elevated privileges to subvert these protections (Li et al., 2023).

The ongoing challenge is that client-side integrity checks can ultimately be compromised by a determined adversary with control over the device's operating environment.

The risks extend beyond location spoofing to fundamental flaws in data protection, such as insecure storage of sensitive information. The following risk flow exemplifies a critical vulnerability:

Step 1: A user logs into a mobile social application.

Step 2: The application insecurely stores the user's session token or credentials in an unencrypted form, such as in plain text within Shared Preferences (Android) or a plist file (iOS) (OWASP, 2021).

Step 3: The phone is either lost/stolen or becomes infected with malware.

Step 4: An attacker can then extract this data through physical access using tools like Android Debug Bridge (ADB) or via the malware that has been installed (ENISA, 2023).

Step 5: Using the stolen session token, the attacker gains unauthorized access to the user's social media account, leading to potential identity theft, privacy breaches, or further social engineering attacks (Spoof et al., 2024).

Location cheating against GPS verification remains a significant threat to Location-Based Service (LBS) applications. These applications typically rely on the client-side application on a smartphone to provide GPS readings. An attacker can subvert this process by blocking the legitimate GPS signal and feeding falsified location data to the LBS application, thereby deceiving the server into believing the device is at a spoofed location (Zhang et al., 2023).

I. Attacks via GPS APIs and Wi-Fi Positioning:

The open nature of mobile operating systems like Android allows attackers to modify GPS-related APIs to return fabricated data from a source other than the phone's hardware GPS module (Li et al., 2023). Furthermore, Wi-Fi Positioning Systems (WPS), used to enhance location accuracy indoors, are also vulnerable. Spoofing attacks against WPS involve an attacker broadcasting beacon frames containing the MAC addresses of access points from a distant location (Location A), thereby tricking a device at another location (Location B) into falsely reporting its position (Zhu et al., 2024). The localization process, which involves a device scanning for APs, querying a location lookup table (LLT) server, and receiving a position estimate, can be wholly compromised by this data poisoning.

II. Advanced GPS Spoofing:

Beyond simple API manipulation, sophisticated attacks target the GPS signal and receiver stack itself:

GPS Data Level Attacks: Modern spoofing can manipulate the navigation message data itself, enabling attacks that go beyond simple position manipulation. For instance, such attacks can cause permanent desynchronization in critical infrastructure like Phasor Measurement Units (PMUs) used in the smart grid (Nighswander et al., 2022).

GPS Receiver Software Attacks: GPS receivers are embedded computers running complex software stacks, making them vulnerable to remote exploitation through their network services. These vulnerabilities are particularly dangerous as receivers are often overlooked in patch management cycles (Wu et al., 2021).

III. Attacks via Server-Side APIs:

The massive reliance on APIs has expanded the attack surface. Server-Side Request Forgery (SSRF) vulnerabilities in LBS APIs are a critical risk. An attacker can exploit an API that fetches a remote resource without proper validation, tricking the server into making unauthorized requests to internal systems, potentially bypassing firewalls to execute location checks (Salt Security, 2024). This flaw is now a top concern in the OWASP API Security Top 10 (2023).

IV. Vulnerabilities from Smartphone Manufacturers:

A lack of transparency and timely security updates from smartphone manufacturers leaves users exposed. Research indicates that many devices run outdated firmware with known, unpatched vulnerabilities, creating a large attack surface for data extraction and device compromise (Sokol et al., 2023). Furthermore, the official device emulators provided to

developers for testing contain fully functional, scriptable simulated GPS modules. These emulators are easily mastered and provide a low-barrier method for attackers to develop and test location spoofing applications without needing physical hardware (Chen et al., 2021).

The core vulnerability in location-based services stems from an architectural flaw that inherently trusts unverified client-side location data, enabling widespread spoofing attacks. Attackers readily exploit mobile operating systems through mock location apps, API manipulation, and debugging interfaces to inject false coordinates that bypass basic client-side defenses. These threats extend beyond simple spoofing to include insecure data storage, where plaintext credentials enable account takeover, and sophisticated wireless attacks that poison Wi-Fi positioning systems. Furthermore, advanced GPS spoofing can manipulate navigation data itself, threatening critical infrastructure, while server-side API vulnerabilities like SSRF allow attackers to bypass all client-side checks. Compounding these issues, manufacturers' slow security updates and accessible device emulators lower the barrier for developing attacks. Consequently, securing location services requires a fundamental shift from client-side trust to robust server-side verification, cross-system corroboration, and comprehensive vulnerability management across the entire technology stack.

2.3.1.3 Eavesdropping

Eavesdropping attacks, also known as snooping or sniffing, remain a pervasive cybersecurity threat, increasingly targeting sensitive personal and business information transmitted over wireless channels. Attackers continue to exploit unsecured or public Wi-Fi networks, employing sophisticated data sniffing techniques and deploying malware to intercept data. This underscores the critical need for robust end-to-end encryption, secure communication

infrastructures, and enhanced user digital literacy to protect against unauthorized data interception (Khan et al., 2022; ENISA, 2023).

The interception of a user's identity in mobile and social network systems remains a critical vulnerability. Once a social network identifier or authentication token is captured, it can be weaponized in replay and spoofing attacks. In such an attack, a malicious actor impersonates the compromised user by replaying the intercepted credential to other devices or services (Zhang et al., 2023). This replay mechanism is also fundamental to wormhole attacks in wireless ad-hoc and IoT networks, where transmissions are captured at one location and tunneled to another to disrupt network routing and create a false topology (Goyal et al., 2021).

Active Adversaries: Man-in-the-Middle (MitM) attacks are a realistic threat on both local and wide-area networks, where an adversary interposes a malicious proxy between clients and servers. This threat model aligns with the long-established Dolev-Yao adversarial model, which assumes total control over the network channel, and continues to be relevant for analyzing the security of modern cryptographic protocols (Bhargavan et al., 2021).

Passive Adversaries: These adversaries covertly monitor and log all network traffic to and from a target. Even when message content is encrypted, a passive adversary can perform traffic analysis to map social connections, determine communication patterns, and infer sensitive metadata, which can be as revealing as the content itself (Nithyanand et al., 2020). The logging of encrypted data also enables future cryptanalysis should the encryption be broken or keys be compromised.

Service Providers: For messaging systems reliant on centralized infrastructure (e.g., traditional Signal server model, Matrix), the service provider is a inherent trusted entity that manages user data. This centralization creates a single point of trust, meaning the service operators could potentially become insider threats, capable of compromising user privacy or being compelled to hand over data to authorities (Álvarez et al., 2024). This risk has accelerated research into decentralized and federated architectures that minimize the trust required in any single provider.

Eavesdropping remains a critical threat where attackers intercept data over wireless channels, employing passive sniffing on unsecured networks or active Man-in-the-Middle (MitM) attacks to position themselves between clients and servers. These attacks can capture sensitive identity tokens, enable impersonation and replay attacks, or perform traffic analysis to infer sensitive metadata from encrypted communications. Furthermore, centralized service providers present an inherent risk as potential insider threat.

The table 2.1 outlines specific defense strategies for both developers and end-users to mitigate these risks, while table 2.2 summarizes on the strategies that can be used to mitigate the specific attack vectors on mobile security Vulnerabilities.

Table 2.1: Defense strategies for both developers and end-users

Threat	Defense Strategy for **Developers**	Defense Strategy for **End-Users**
General Eavesdropping and MitM	Implement robust End-to-End Encryption (E2EE) for all data in transit and at rest. Use certificate pinning to prevent impersonation of servers.	Avoid unsecured/public Wi-Fi for sensitive activities. Always use a reputable VPN on untrusted networks to encrypt all traffic.
Credential Interception and Replay Attacks	Utilize short-lived, cryptographically secure authentication tokens. Implement token binding and anti-replay protections (e.g., nonces, timestamps).	Use multi-factor authentication (MFA) everywhere to mitigate the impact of a stolen password or token.
Traffic Analysis	Pad messages to a uniform size and use cover traffic to obscure real communication patterns, making metadata less useful to an adversary.	Be aware that even with encryption, who you talk to and when can be revealed. Use services that prioritize metadata protection.
Service Provider as Insider Threat	Adopt decentralized or federated architectures (e.g., based on Matrix protocol) that distribute trust and do not grant a single entity full data access.	Choose communication services with a proven E2EE model and a transparent privacy policy that minimizes data retention.
Traffic Analysis	Pad messages to a uniform size and use cover traffic to obscure real communication patterns, making metadata less useful to an adversary.	Be aware that even with encryption, who you talk to and when can be revealed. Use services that prioritize metadata protection.
Service Provider as Insider Threat	Adopt decentralized or federated architectures (e.g., based on Matrix protocol) that distribute trust and do not grant a single entity full data access.	Choose communication services with a proven E2EE model and a transparent privacy policy that minimizes data retention.

A comprehensive defense requires a shared responsibility model. **Developers** must build secure, privacy-by-design systems with strong encryption and minimal data collection, while **Users** must adopt secure practices like using VPNs, enabling MFA, and making informed choices about the services they trust with their data.

Table 2.2: Mobile Security Vulnerabilities, Attacks Vectors, Risk Level and Mitigation Strategies

Vulnerability Category	Specific Attack Vector / Technique	Key Impact / Consequence	Risk Level	Mitigation Strategy
Fundamental Architectural Flaw	Trust in client-side location data without server-side verification.	Enables all subsequent location spoofing attacks; foundation of the security problem.	**Critical**	Implement **server-side location corroboration** using independent signals (e.g., IP geolocation, cell tower triangulation) and behavioral analytics to flag impossible travel.
Client-Side Data Manipulation	1. Use of mock location apps. 2. Modification of GPS APIs. 3. Exploitation of debugging interfaces.	Falsified coordinates are accepted by the application server, enabling location cheating.	**High**	Deploy **sensor fusion analysis** on the server-side (e.g., checking accelerometer/gyroscope data against GPS track). Detect and block requests from rooted/jailbroken devices or when mock location settings are active.
Insecure Data Storage	Storing session tokens/credentials in plaintext (such as Shared Preferences, plist files).	Account takeover via physical access or malware, leading to identity theft and privacy breaches.	**High**	Use operating-system-backed **secure storage** (Android Keystore, iOS Keychain) for all sensitive data. Implement token binding and short-lived session tokens.
Wi-Fi Positioning System (WPS) Spoofing	Broadcasting beacon frames with spoofed MAC addresses from a different location.	Tricks the device and server into reporting a false location, compromising WPS reliability.	**Medium**	**Cross-verify** WPS-derived location with GPS/Cell-ID data. Service providers should maintain and constantly audit WPS databases to identify and remove poisoned access point data.
Advanced GPS Spoofing	1. Data-Level Attacks: Manipulating the GPS navigation message itself. 2. Receiver Software Attacks: Exploiting vulnerabilities in the receiver's OS/software.	Causes severe damage beyond simple spoofing (e.g., crashing receivers, desynchronizing critical infrastructure like smart grids).	**Critical** (for infrastructure) **Low** (for consumer apps)	For critical systems: Use **cryptographically secure GPS signals** (e.g., M-Code). Deploy **multi-source timing and location** (e.g., combining GPS with eLORAN, inertial navigation). Enforce strict **patch management** for all receivers.
Server-Side API Exploitation	Server-Side Request Forgery (SSRF) in LBS APIs.	Bypasses client-side checks; allows attacker to trick the server into executing unauthorized location checks from within the network.	**High**	Implement strict **input validation** and **allow-list** for all user-supplied URLs. Employ **network segmentation** to restrict the application server's ability to connect to internal systems.
Vendor/Manufacturer Issues	1. Lack of timely security updates and transparency. 2. Availability of official device emulators with scriptable, fake GPS modules.	Creates a persistent attack surface for data extraction and lowers the barrier for developing and testing spoofing attacks.	**Medium**	**Defense-in-Depth:** Assume client-side can be compromised. On the server side, implement policies that distrust data from outdated OS versions. For emulator detection, analyze sensor fingerprints and hardware parameters that are difficult to perfectly emulate.

2.3.2 Mobile Social Networking Attack Techniques

There are two key layers of information security such as technical and human layer of information security. Such strategies are present in the technical layer such as use of firewalls, software patches, among others, there are intrusion detection systems, anti-virus programs and encryption. Nevertheless, one significant information security defense layer that is not provided with the necessary attention is the human layer. The arguably the most significant layer of defense of information security is the human element. When a malicious party can penetrate the human aspect well, then he or she can do away with all the other aspects of the defense against malicious attacks. When the human layer is violated, the time, money and resources that were used to protect the other layers of information security defense have gone to waste. Thus human layer needs most attention as it is involved in the launching of attack and the application of the techniques of prevention of attack at the technical layer. The different techniques of attack are as given in the following subsections.

2.3.2.1 Social engineering

Social engineering is a concept that includes all sorts of malicious practice that include phishing, pretexting, baiting, quid pro quo, and tailgating (piggybacking). Having this human-centered attitude in mind, it is upon a user to be mindful of the malicious actors that practice social engineering attacks in search of human greed and ignorance (Weichbroth, 2020).

Social engineering refers to the process of influencing, controlling, or deceiving a person to disclose sensitive information, get unauthorized access, or commit fraud by linking to the person to develop confidence and trust.

The most frightening aspect of social engineering is that it is a human error-based method as opposed to software and operating system vulnerabilities. Errors of legitimate users are significantly lesser.

Social engineering involves the application of any set of methods to achieve access to otherwise protected space or information. Social engineering is available in various forms. Phishing is one of the more popular forms that have been covered by the press. False correspondence is known as phishing to obtain information. The phishing attack may be in the shape of an email by a fraudulent email writer, a spoof site, or even a fake telephone call involving a bogus Instant Voice Response system. One of the key phishing cases is the receipt of a mail that can be said to be a large corporation like eBay or PayPal.

The email will deceive the recipient, that there is an issue with their eBay or PayPal account and will seek to redirect the recipient to a rogue site or phone number to solve the issues. When the recipient clicks on the sporesite or dial the phone number, he or she will be requested to enter some sensitive details such as their social security number or account usernames, passwords and pin numbers under the pretext of confirming their details (Emekauwa, 2007).

Pretending is another type of social engineering. Pretexting is the effort to gain access to personal information or a restricted area by developing counterfeit pretexts. Telephone pretexting is common and mainly, entails use of impersonation tactics whereby the attacker poses to be what he is not. A pretexting situation would be a call one gets at work by a person

who is faking to be a support tech at the helpdesk IT and that he had a virus on your computer system that they will need to fix, but that they require your username and password to do so. The other example would be the IT helpdesk of an organization when the user makes a call in which he claims to be a user who has lost his/her email password and wants to be reset. An experienced pretexter will employ different reasons to cause a feeling of urgency or pressure in his request including pretending that he is a company vice president or that he has a meeting to attend in a few minutes that he must have access to his email (Emekauwa, 2007).

Spoofing

Spoofing in computer world is considered as stolen identity where an individual masquerades as someone else, an organization or business enterprise with an aim of accessing confidential personal information such as user names and passwords, bank account details, and credit card details. The spoofing is not only included in the arrangement of phishing but also a strategy of acquiring access directly to the computer or computer network of a person or organization. Some known types of spoofing include: IP spoofing, URL spoofing, Email spoofing, DNS spoofing and MAC spoofing.

Undoubtedly, smartphones have become one of the most popular technologies in the present day. The mobile positioning and navigation services are being used daily by billions of users in the world (Wahlström, Skog, & Hände, 2017). The uses would be in the navigation of our map, in recommending services, and in the social media, among others, where precise and authentic location information is needed. Meanwhile, these location-based applications are a honey pot to the attackers since they have become very popular. The attackers who do this

hope to make use of the location information, which is supplied by smartphones, to commit crimes. The traditional positioning system before depends on the GPS signals. But as the WiFi technology advanced at a high rate, WiFi-based positioning system was extensively implemented not only as a supplementary location-based service of GPS, but also as a dominant positioning service (Bell, Jung, and Krishnakumar, 2010).

The current positioning system is not resistant to spoofing attacks whereby the attackers can use falsified wireless signals to alter the smartphone location result as most positioning signals are not encrypted. The earlier studies indicate that a simple attack that compromises with the smartphone location system to an inaccurate destination is simple to attain. However, dynamic attack where the smartphone is made to think that it is traveling on the artificial path that was created by the attacker is restricted (Zeng, Shu, Liu, Dou, and Yang, 2017).

Phishing

Phishing is a form of deception in which an attacker attempts to fraudulently acquire sensitive information from a victim by impersonating a trustworthy entity (Miller, Katelin, Zhang, & Terwilliger, 2020).

According to Bose & Leung (2009), the phases of a phishing attack include: (1) preparation: identify target companies for phishing and explore potential system vulnerabilities; (2) mass broadcast: spread phishing message to the public; (3) mature: wait for victims to respond to phishing messages; and (4) account hijack: conduct identity theft to cause financial loss to victims. Berghel, (2006) maintains that the essential requirements of effective phishing are as follows: (1) look real; (2) present itself to an appropriate target-of-opportunity; (3) satisfy

the reasonableness condition; (4) cause the unwary to suspend any disbelief; and (5) clean up after the catch (Miller, Katelin, Zhang, & Terwilliger, 2020)

Despite best efforts, current automated prevention techniques fall short and quickly become outdated (Jensen, Dinger, Wright, & Thatcher, 2017). Therefore, new strategies should be implemented to address this need. There is much to be learned from current research on phishing prevention and this information can be consolidated into a three-step plan of action for managers and end-users to protect themselves and their businesses (Miller, Katelin, Zhang, & Terwilliger, 2020).

2.3.2.2 Direct anonymity

The attacks are directed at identity and privacy of the users. Direct anonymity attack attempts to violate the anonymity of the users by exposing information and location of the users as they access a social network site using mobile devices. As discussed, devices in the same system are able to discover the name of SN user that will directly infringe privacy of the user (Racha et al., 2011). The mobile social network systems through the information exchange model offer minimal security to the privacy of the user. These systems entail the user granting access to his or her information on the social network profile and at the same time connecting the same information to the user as it appears. As an example, Facebook app will normally insist on the user to accept that the application is authorized to gain access to his/her information in Facebook API, inseparably connecting such information to the identity of the user (Beach et al., 2009). In other social networks like WhozThat and Social Aware systems, any person close to the mobile user can snoop a social network ID being shared by a user or

even eavesdrop on information sent by an individual openly over a wireless connection since no encryption is applied, albeit loosely, to data transferred by the link layer (Berker, 2007).

In client-server mobile social network systems, direct anonymity attacks may also occur. Although the social network IDs of the users are not typically directly exchanged among the mobile devices in such systems, mobile or fixed devices may still track a user by recording the date and time when they were found within the vicinity of the device(s). As every device in these systems may locate the social network user names and the frequently-full names of the close users, the privacy of the latter may be violated. Therefore, we get a direct problem of anonymity- the revelation of user names and locations in client-server systems permits the anonymity of the user to be violated (Beach et al., 2009).

When an attacker tries to access the users identify and other details by accessing unauthorized means to the social network ID of the user. This problem can be seen in both user-server domain and user-user domain. By getting any single information of the user the other related information of the user can be easily cracked by the attackers.

2.3.2.3 Indirect anonymity

Indirect or K-Anonymity attack occurs when a number of pieces of information non compromising in itself indirectly jeopardize the identity of users by assembling the related pieces of information to trace back to the identity of the users. As an illustration, list of favourite stores and restaurants of a particular user can easily be used to trace a user and determine his/her whereabouts (Racha et al., 2011). One to be considered as a worthy challenge is the one of sustaining complicated mobile social networking applications that

have personal information without interfering with the anonymity of the information provider. Although the user may not explicitly give his/her identification information, the information that he/she gives about his/her social networks (including preferences) can be traced to that of the user through the social network site or data stored in mobile and stationary devices in the environment. The indirect anonymity issue is the one whereby a bit of information indirectly violates the identity of a user. This can be illustrated by the case of a user-specific piece of information being conveyed out, say a list of favorite movies of the user, this information can easily be traced to the user. K-anonymity problem arises when one has n pieces of information or n groups of information that can be jointly utilized to a user identity (Beach et al., 2009). Here, given that a group of data can only be joined to a group of k or less groups of users, the anonymity of the user is still violated to some extent associated with k. Thus, it is a question of creating an algorithm that is capable of determining what is and is not to be released to ensure the anonymity of the related users.

The attackers attempt to determine or map the identity of the user by hacking any N number of information about the user and this uniquely maps the user. This issue can be regarded in user-server and user-user areas as well. The answer to this problem is to develop an algorithm that exploits a minimal set, which consists of set of information of the user that cannot map to the user directly, but rather, maps to K or fewer users. In this way, the attackers will be driven to the stage of confusion when it comes to discovering the identity of the original user.

Eavesdropping is a major risk in which intruders capture data through wireless networks, either passively sniffing an unsecured network or actively through Man-in-the-Middle

(MitM) attacks in order to place themselves between the clients and the servers. Such attacks may steal sensitive tokens of identity, facilitate impersonation and replay attacks, or conduct traffic analysis to deduce sensitive metadata of encrypted messages. Moreover, centralized service providers are a potential risk factor of insider threat.

2.4 Theories, Models and Frameworks

The terms "theory," "model," and "framework" represent distinct but interconnected tools that structure research, guide analysis, and contribute to the body of knowledge. Understanding their hierarchy and function is crucial for rigorous scientific practice.

The terms are sometimes used loosely; their precise meanings are vital. A theory explains *why*, a model shows *how*, and a framework guides *where to look*. Together, they form the essential scaffolding of scientific reasoning, enabling researchers to move from abstract concepts to empirical testing and cumulative knowledge advancement.

2.4.1 Mobile social network security techniques

Mobile phones, laptops and tablets have been enjoying exceptionally explosive growth in terms of wireless communication, computing and sensing. Indicatively, over 30 cell phones are sold in the world every second. Besides this, the nascent industrial sensor networks are not only economically but also strategically significant. Moreover, wireless security provides a set of objectives and requirements that is technically difficult (Sheng andMiodrag, 2012).

Handshake Scheme (Mobile Healthcare Social Network Application). After the successful same-symptom based handshake scheme (SSH), the patients Ui and Uj are able to share their mutually protected health information (PHI) and experiences with each other on a safe basis, and support and motivate each other. MHSN can be embraced by patients in large portions, because of these promising functionalities. Moreover, due to the unavailability of Access Point (AP) to a patient in the mobile setting, active patients, according to the same-symptom-

based social relationship can also assist each other in communicating his PHI. Through this, the PHI reporting delay may be minimized.

Semantic security. In order to satisfy the need of SSH scheme, identity-based encryption (IBE) must be semantic security (indistinguishable) against selective-PID-Symptoms and chosen plaintext attacks. Specifically, consider an adversary A is first given the public parameters and selects the specific PID pid_ and symptom T_ in advance. Then, A is allowed to access two types queries to the key generation oracle OK: i) query for the challenged pid_ on other symptoms T, where T _= T_; and ii) query for other pid, where pid _= pid_, on the challenged T_. At some point, A outputs a pair of nonce N0,N1∈ GT . Then, after one nonce Nb, b ∈ {0, 1}, is encrypted with the challenged pid_, T_, the adversary A must decide which nonce has been encrypted.

2.4.2 Attack Prevention Techniques

The issues of security and protection differ in an application on an online social network. However, the user desires to make new friend in order to enhance his or her social network as well as to be informed by different people.

Considering the Facebook ID of a user, it is easy to view the public information of the user and estimate the position of the user. This type of system is compatible with the current Facebook application model and it has a grave and noticeable privacy issue (Beach et al., 2009).

In the whozthat and socialAware systems, any person within the vicinity of the mobile user can sneak a user associated social network ID or spy on data that is transmitted openly over

a wireless connection since all data that is exchanged over the wireless connection is not encrypted. Social networking IDs are easily exchanged in clear-text formats in systems like whozthat and socialAware with the result of creating unacceptable security and privacy risks.

In Secure SocialAware Architecture though, this architecture does not mean the implementation of any given component. As an example, the EIDs of can be shared using various wireless/ cellular protocols rather than the Bluetooth. They might also be operated with the help of a centralized infrastructure in the Internet or being part of a peer-to-peer trust network. This is still to be developed as alternative implementations in future. The preference selection and delivery mechanism to the SC might be streamlined towards various measures. Specifically this mechanism may be created to grant k-anonymity to a group of users EIDs and not a single one. The issue is connected to a larger problem in set theory where a set of social network data connected to a set of users is selected where the set of preferences cannot be mapped onto any or any of the users in some guarantee.

With user social networking ID and access, another user can access the public information of a user in a manner that they may not have intended by merely looking at the open profile of the user on asocial network sites.

2.4.3 Security Frameworks in Mobile Social Networking

A security framework outlines regulations and protocols towards developing nan upholding security control (security.com). Frameworks explain procedures that are applied to safeguard an organization against cybersecurity threats; they assist IT specialists maintain their organization at par and shield it against cyber attacks.

There are four types of frameworks

Control:

Network and Work Station Controls

i. Create a security team with a basic strategy

ii. Identify a baseline set of controls

iii. Perform a gap analysis

iv. Prioritized control implementations

NIST 800-53B is an example of a control security framework.

It focuses on four baselines;

i. Low impact

ii. Moderate impact

iii. High impact

iv. Privacy control

Anothe3r example of control framework is CIS (Center for Internet Security) Benchmarks, which has 18 critical controls

Inventory and Control of Enterprise Assets

Identify and manage all devices connected to your network.

i. Inventory and Control of Software Assets

   Track and manage all software to ensure only authorized software is installed.

ii. Data Protection

   Protect organizational data through data classification, handling, and encryption.

iii. Secure Configuration of Enterprise Assets and Software

   Apply and maintain security configurations on hardware and software.

iv. Account Management

   Manage access credentials and privileges for users, administrators, and service accounts.

v. Access Control Management

   Ensure users have the least necessary access, and that access rights are regularly reviewed.

vi. Continuous Vulnerability Management

   Continuously acquire, assess, and take action on information regarding vulnerabilities.

vii. Audit Log Management

   Collect, alert, review, and retain audit logs to detect and understand attacks.

viii. Email and Web Browser Protections

   Protect against phishing and other web-based threats.

ix. Malware Defenses

Control installation, spread, and execution of malicious code.

x. Data Recovery

Ensure the ability to restore data after a loss or ransomware attack.

xi. Network Infrastructure Management

Secure network infrastructure devices (e.g., routers, switches) and manage network architecture.

xii. Security Awareness and Skills Training

Train users to recognize and avoid cyber threats.

xiii. Security Operations Center (SOC) Logging and Monitoring

Collect and analyze security-related data to detect and respond to threats.

xiv. Service Provider Management

Evaluate and monitor the security practices of third-party service providers.

xv. Application Software Security

Ensure applications are securely developed and maintained.

xvi. Incident Response Management

Establish and practice a plan to respond to security incidents.

xvii. Penetration Testing

Regularly test the effectiveness of defenses using simulated attacks.

A mobile app security framework provides a structured approach to designing, developing, and testing secure mobile applications. It outlines best practices, guidelines, and standards

to mitigate risks and protect user data. Key frameworks include OWASP MASVS, OWASP MSTG, and NIST CSF.

Most of the literature has focused on the role of the Internet and computers in developing and maintaining relations. Mobile telephony provides an increasingly important communication channel in facilitating social connections, which are the focus of the research work.

2.4.3.1 WhozThat

WhozThat is an answer system to the simple social question of Who is That. WhozThat is a combination of the existing online social networks and the Smartphone mobile device, and a platform to develop a new breed of applications which are context-aware and use social network information. Social networks give access to a large amount of personal data about individuals such as name, gender, contact, interests, music preference, movie preference, book preference and friendship networks with other users, which are only some of the areas where users post information on their social network profiles (Aaron Beach, Mike Gartrell, Baishkhi Ray, Richard Han, 2009). WhozThat is based on the peer-to-peer model of sharing the social network identifiers of the users across smartphones and other devices, with social networks identifiers being delivered through a wireless system like WiFi or Bluetooth.

WhozThat prototype platform was used, where the plain-text display of the Facebook ID of the user was advertised over Bluetooth. As the user walks into a new room his/her mobile phone notices the list of Bluetooth devices in the room and requests each device to give the Facebook ID of its user. The ID is then used to get the information of each user in Facebook profile using the facebook API web service via Internet connection. This allows the user to

have access to whatever information he was allowed to access under the use of Facebook without violating the Facebook privacy policies. Nevertheless, even staying within the privacy regulations of Facebook does not leave the privacy of the user intact (Beach et al, 2009).

2.4.3.2 Secure SocialAware Architecture

The SocialAware is a context-aware mobile social networking service construction framework within a ubiquitous computing context (Gartrell, 2008). The SocialAware apps rely on the identity sharing protocol of WhozThat to exchange social network IDs of users between mobile and fixed devices (Beach et al., 2009). SocialAware is contrasted with WhozThat as S: Where the mobile client (also known as the Mobile Component or MC) of the mobile device) is an information provider to the stationary system (also known as the Stationary Component or SC). The data as supplied by each Mobile Component is the social network ID of the user and information as supplied by the Stationary Component is the information about the social network profile of the user as pertains to his/her social network ID. To identify which actions to undertake depending on the preferences of the user, the SC uses one or more domain-specific web service recommender systems. The Netflix REST API web service is one such recommender system. This system may be regarded as a simplistic representation of the fundamental elements of any context-aware system that uses social networks where the context represents the location of the user and user personal information that needs to be shared with a central system.

Secure SocialAware (SSA) architecture comprises of three key components stationary component (SC), mobile component (MC), and authentication server (AS). The SC is

integrated with the environment of the user whereas the MC is found in the mobile devices. The SC is involved in monitoring the existence of users, getting the information of the users on the AS and taking actions on the basis of the information. The peer on the hand of the mobile phone belonging to a user encrypts the user identity (EID) and then advertisement on the identifier is done by the MC via a wireless communication technology like Bluetooth in a manner that it is accessible to the SC. The AS makes an EID of every MC and offers a service to the SC to allow the SC to get the information of the user in a social networking web site provided an EID is provided.

2.4.4 Theoretical Framework

The security model which is a description of the way security is implemented is one of the fundamental concepts in computer and information security. A security model is a statement representing the requirements that are intended to support and implement a particular security policy appropriately. Computer and information security encompasses lots of fields in an enterprise. Any place has security weaknesses and hopefully, it does have countermeasures that elevate the level of security and offer improved security. The main attributes of information security include the following; Availability, Integrity and Confidentiality.

In the development of the Internet, a lot of developers were busy in creating new protocols of instant messaging (IM), which would bridge individuals over the network. In the present world, there are myriads of IM protocols being transmitted in every corner of the world that are capable of relaying data between one end and the other across the network, either in a secure way, utilizing the cryptography tools, or in an unsecured manner that is usually in the

plain test format. In order to give a general overview, we chose to show some of the diametrically opposite protocols of less secure protocols (i.e. the IRC protocol in its original form) to highly security-sensitive solutions (i.e. the Signal Protocol).

In the Mobile-based social networks there is no contrast in the sense that they experience issues of vulnerabilities which must be addressed and secured to achieve privacy and integrity.

2.4.4.1 Security Models

Computer security model is a plan of security policy specification and implementation. A security model can be based on a formal model of access rights, a model of computation, a model of distributed computing or no specific theoretical basis at all. A computer security policy is used to implement computer security model.

2.4.4.1.1 The Signal Protocol

Whisper Systems started the Signal Protocol, which released two apps RedPhone and TextSecure in the year 2010, the two apps aimed to provide secure calling and texting respectively and were proprietary in nature. Coming in the year 2011 it was acquired by Twitter who initially moved to close down the services. After public outcry, the company released the apps under the GPLv3 open-source license.

In 2012 the founder of Whisper Systems, Moxie Marlinspike, left Twitter. He founded a new company Open Whisper Systems which, aimed at continuity of developing the two apps. In 2014 the company would release an app for iOS called Signal that supported the RedPhone calling protocol. Later the TextSecure Protocol would be ported to iOS and merged into the

Signal app as well. Around the same time the RedPhone and TextSecure apps for Android were merged into a single app also called Signal. To complete the rebranding the TextSecure Protocol was rebranded as the SIgnal Protocol.

The Signal Protocol is developed by Open Whisper Systems, aimed to create a secure messaging protocol for use in the modern world. The protocol is used in their own app, Signal, along with WhatsApp, Facebook Messenger, and Google Allo. The Signal app and WhatsApp both use the protocol for all messages, whereas Facebook Messenger and Google Allo only use it for "secret messages." Both WhatsApp and Facebook Messenger claim to have over a billion users. WhatsApp is especially significant since all messages are encrypted with the protocol. (Katriel, Cas, Benjamin, Garratt, & Douglas, 2017) The protocol was originally released in 2013 under the name of the TextSecure Protocol. It was later renamed to the Signal Protocol. After its original release an update was made in 2014. The second version brought in several new features and began the use of the Axolotl ratchet.

2.4.4.1.2 End to End (E2E)

The adoption of end-to-end encryption(E2EE) in a messaging service implies that only the sender and the recipient of a specified message can access the contents of this message. In the absence of E2EE, your message can be encrypted when it is being transmitted to the server, however, the server can probably read it. Under E2EE, message is always encrypted and this is how it passes through various intermediaries that can be any possible.

Signal, an app for messaging, created by Open Whisper Systems (OWS), is becoming more and more popular with each day among individuals across the globe as an app that offers

end-to-end encryption to its users. It communicates through the Signal Protocol, an open-source end to end encryption protocol, which is also created by OWS.

Today, the use of end-to-end encrypted messaging has gained popularity, and large proprietary applications like WhatsApp or Facebook Messenger have started to implement end-to-end encrypted messaging, and the desire to maintain the privacy of communication evoked interest due to the Snowden revelations. In end-to-end encryption, the server where messages are stored on behalf of a user or any other third-party enemy intercepting data, as the message is being sent, is not able to see the content of the message since it is being encrypted. The term end in the end-to-end encryption can be understood as the endpoint, which, in this situation, is the client device of the user and not the server (Ksenia, Francesca, and Harry, 2016).

2.4.4.1.3 Lightweight Cryptography

Networked computer system security, (as confidentiality, integrity and availability) is not a new issue but in the case of Embedded Systems (ESs), its diverse intrinsic and frequently application specific nature makes security methods used in personal and enterprise systems inadequate or even irrelevant. These features are characteristic of resource constraints (in particular, computational capabilities, memory and power), dynamically developed and remotely controlled or even uncontrolled networking and performance in hostile environment and time-constrained applications (Manifavas, Hatzivasilis, Fysarakis, and Rantos, 2013).

We cannot imagine the life of an average man without numerous gadgets that we possess in the present day. A great number of homes employ devices having an inbuilt operating system

(other than common personal computers), capable of being linked to the Internet, and even formed into a wireless network. People are everywhere enclosed by a number of terminals, readers, sensors among others. This type of proliferation of smart technologies is significant in increasing the issue of data security. Nevertheless, at this point, it cannot be proposed to have a cryptographic primitive that can be executed in any kind of target device. We know that AES is a very powerful algorithm and is well performing. Use of AES is certainly recommended in high-end equipment, in a vast range of embedded systems or some low-end equipment (with a number of restrictions). Nonetheless, common cryptographic algorithms cannot be applied in particular devices that have very limited resources (Sergey and Sergey, 2011).

These are RFIDs; low-end smart cards (including wireless); wireless sensors; indicators, measuring devices, custom controllers among others. Thus, the implicit assumptions and strategies of the design of algorithms that are supposed to be utilized in devices with immensely low resources differ slightly with the design criteria of popular cryptographic algorithms. The lightweight cryptography is a branch of modern cryptography that covers this very specific field. Lightweight cryptography does not dictate particular properties of identifying a cryptographic algorithm as lightweight, but the usual characteristics of lightweight algorithms are very high demands on necessary resources of target hardware such as: Size necessary to implement hardware; computational power of microprocessors or microcontrollers; RAM; Read-only memory(ROM) among others (Sergey and Sergey, 2011).

All lightweight cryptography designers are forced to trade off on the cost, performance and security. Combining two out of the three design aims security and cost, security and

performance, or cost and performance is often straightforward, but the combination of all the three design aims is extremely challenging. An example is a pipelined, side-channel-resistant architecture can be a safe and high-performance hardware implementation, which has a high area requirement, and hence high costs. Conversely, one can come up with a safe and cheap hardware implementation with the disadvantage of low performance (Thomas, Sandeep, Leif, Christof, and Axel, 2007).

2.4.4.1.4 Blowfish algorithm

Blowfish is a well-known security algorithm that is lightweight and was invented by Bruce Schneier during the year 1994. The algorithm operates on the same line as DES and it uses block coding that has block sizes of 64bit. Blowfish is a symmetric algorithm in encryption i.e. it encrypts and decrypts the messages using the same secret key. Blowfish is also a block cipher, which implies that it breaks up message in an encrypted and decryption process into fixed length blocks. Blowfish has a block length of 64 bits; messages that are not multiple of eight bytes need to be padded (Valmik & Kshirsagar, 2014)..

Table 2.3 Comparison of Algorithms

Algorithms	Key size	Block size	Round	Level of Security	Encryption speed
DES	64 bits	64bits	16	Not secure enough	Very slow
3DES	112 or 118 bits	64 bits	48	Adequate security	Very slow
AES	128, 192, 256 bits	128 bits	10,12,14	Excellent security	Faster
RC4	Variables	40-2048 bits	256	Adequate security	faster
RC6	128-256	128 bits	20	Good security	Average

BLOWFISH	32-448	64 bits	16	Highly secured	Very fast

Table 2.1 shows that of all algorithms; blowfish algorithm is more secured and have great encryption speed. Blowfish algorithm is the best in terms of execution time, avalanche effect, power consumption, memory usage, throughput, security among another characteristic. In this regard, blowfish algorithm is well suited for the secure data transmission in constrained resources devices.

2.5 Research Gap

The provided literature review extensively catalogs the vulnerabilities and attack techniques in Mobile Social Networking (MSN), such as location cheating, eavesdropping, and social engineering. It also discusses a number of security frameworks and theoretical models, notably the Signal Protocol and Blowfish algorithm as lightweight cryptography.

However, a critical gap emerges between the **technical solutions** and the **socio-technical ecosystem** in which they must operate as in figure 2.1. The text repeatedly highlights that the security of MSN is not just a technical problem but is deeply intertwined with **human factors** and the **centralized, proprietary control of social networking platforms**.

Therefore, there is need to develop an integrated security model for Mobile Social Networking that effectively mitigates technical vulnerabilities while simultaneously addressing the inherent dependencies on social network platform owners and the persistent risks introduced by the human layer. This is comprehensively addressed in chapter five.

The gap is multi-faceted, as illustrated in figure 2.1 the lack of authentication logic

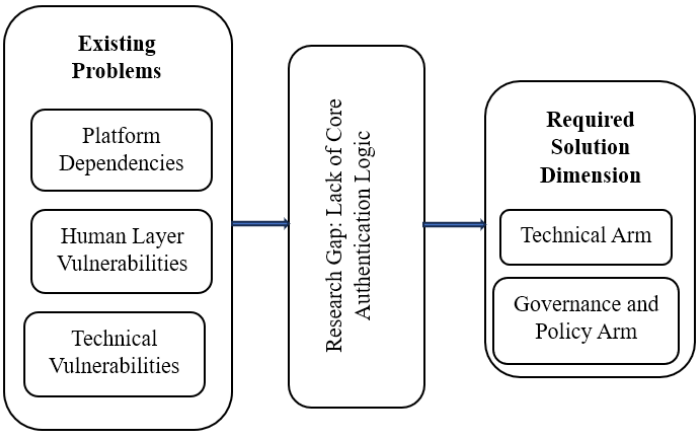


Figure 2.1: Research Gap

Dependence on Social Network Owners and Non-Standardized APIs: According to the text, the platforms that are intended to be defended by technical solutions strangle them. It is a basic limitation that any model suggested should overcome. Existing APIs do not suffice to gather social data on social networking sites because data format and model along with access policies is not standardized... Efforts have been made on open social interfaces but they have since been found not to be practical, primarily due to reliance upon social network owners.

The Critical Role of the Human Layer: The text specifically refers to the human layer as the most significant layer of defense and believes that any efforts to bypass it make all other technical defenses ineffective. However, the majority of the solutions discussed are technical. Human factor is the one that arguably most significantly contributes to information security. When a person with ill motives can effectively evade the human layer,

he/she is able to evade all the other information security defensive layers. The most dangerous thing about social engineering is that it is not based on vulnerabilities of software and operating systems but human error.

The Limitations of Purely Technical Architectures: Even with more secure architecture as suggested by Secure SocialAware (SSA), the text admits that its implementation is subjected to the limitations of the ecosystem and that some of the major issues still remain open. The manner, in which preferences are selected and sent back to SC, might be streamlined to various measures. Specifically in this mechanism could be implemented to achieve a k-anonymity of a group of users EIDs and not individual users. This is an issue associated with a larger set theory issue. This has not yet been considered as possible alternatives in the future. This implies that the existing models are partial and they have not effectively addressed the indirect anonymity (k-anonymity) issue practically.

The current body of literature does a brilliant job at the diagnosis of individual issues (location spoofing, insecure storage, phishing) and suggesting point solutions (encryption, algorithms, particular protocols). It, however, does not give a comprehensive framework that integrates these technical solutions with mechanisms to deal with platform dependency and empower the end-user.

Lack of Authentication Logic:- The absence of a cohesive security model around Mobile Social Networking, which is both effective in reducing the technical vulnerabilities and also in dealing with the underlying reliance on the social network platform providers as well as the ever-present risks presented by the human factor.

2.6 Theoretical Model

The model was derived from the deep analysis of the literature review, with the core components as;

Core Security Objectives

The foundation of any security measure in an MSN is to uphold the three core attributes of information security:

Confidentiality: Ensuring that data (e.g., Protected Health Information, social profiles) is accessible only to authorized parties.

Integrity: Guaranteeing that data has not been altered or destroyed in an unauthorized manner.

Availability: Ensuring that systems and data are accessible and usable when needed by authorized users.

Security Techniques & Cryptographic Foundations

This layer details the specific technical methods used to achieve the core objectives.

Encryption Protocols:

End-to-End Encryption (E2EE): A paradigm where data is encrypted on the sender's device and only decrypted on the recipient's device, preventing intermediaries (like servers) from reading it. The Signal Protocol is a leading example.

Lightweight Cryptography: Cryptographic algorithms designed for resource-constrained devices (e.g., sensors, RFIDs, low-end mobile devices). They optimize the trade-off between security, cost, and performance.

Threat and Vulnerability Landscape

This component identifies the common risks and attack vectors in MSNs.

Eavesdropping & Snooping: Intercepting unencrypted data transmitted over wireless connections (e.g., Bluetooth, WiFi), as seen in early WhozThat and SocialAware systems.

Privacy Leakage via Social IDs: Using a publicly broadcast social network ID (like a Facebook ID) to access a user's profile information in ways they did not intend, potentially revealing location, interests, and connections.

Insecure Data Exchange: Transmitting sensitive data, such as Encrypted Identifiers (EIDs) or social IDs, in plaintext, making them vulnerable to interception.

Device Proliferation: The massive growth of mobile and sensing devices increases the attack surface, making inventory and control of assets a critical first step in defense.

Security Frameworks and Architectures

These are the blueprints and policies for building secure systems and managing security controls.

Control Frameworks (Proactive Management):

CIS Critical Security Controls: Provides a prioritized set of 18 actionable controls. Key controls for MSNs include: **Inventory and Control of Enterprise Assets & Software**

Data Protection (via classification and encryption)

Continuous Vulnerability Management

Account Management and Access Control Management

NIST 800-53B: A control baseline framework categorized by impact levels (Low, Moderate, High), helping organizations select appropriate controls based on risk.

Architectural Models (System Design):

Insecure Architecture (WhozThat): A peer-to-peer model where devices broadcast plaintext social network IDs, leading to inherent privacy risks.

Secure Architecture (Secure SocialAware - SSA): A three-tiered model that enhances security by introducing an Authentication Server (AS) to manage encrypted identifiers (EIDs). This architecture:

Replaces plaintext IDs with encrypted EIDs.

Decouples the mobile component (MC) from the stationary component (SC) via a trusted AS.

Allows for future enhancements like providing k-anonymity for users.

Layer 1. **Foundation:** All security efforts are directed towards achieving the **Core Security Objectives**.

Layer 2. **Implementation:** To meet these objectives, specific **Security Techniques** (like the Signal Protocol for E2EE) are employed, as shown in figure 2.2.

Layer 3. **Context:** The choice of techniques is driven by the need to mitigate the identified **Threats and Vulnerabilities** (such as using E2EE to prevent eavesdropping).

Layer 4. **Governance & Structure:** The entire system is built and managed according to a **Security Framework** (e.g., following CIS controls for device management) and a secure **Architecture** (e.g., the SSA model with an Authentication Server) to ensure a comprehensive and maintainable security posture.

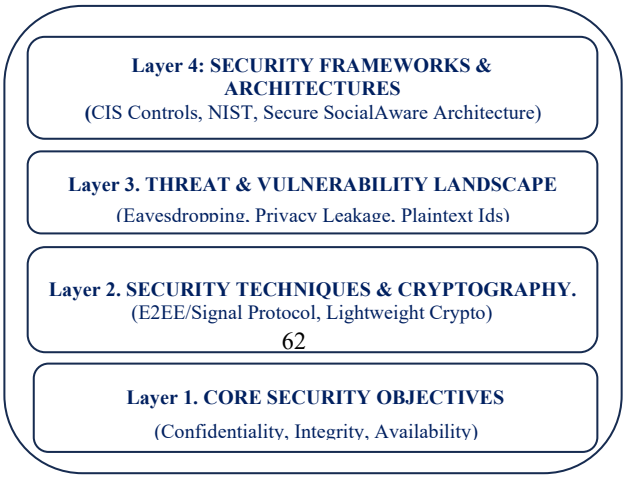


Figure 2.2 Theoretical Model for MbSN

2.7 Conceptual Framework

Beach et al. 2010 state that after the interception of the social network ID of one user within a peer-to-peer mobile social network system, a replay and spoofing attack can be instigated. In a spoofing attack, the attacker can impersonate the user his/her ID was intercepted (the compromised user) by merely transmitting (replaying) the intercepted ID to mobile or stationary devices which require the user to send social network ID. Therefore, the spoofing attack is done using the replay attack in which the ID of the compromised user is repeated maliciously. Therefore the information remains at stake in case a person can retrieve the ID, this test will be in a position to process the security of the information on the server.

The conceptual framework on Figure 2.3 describes the framework for mobile-based social networks and the security prevention attacks are directly dependent on the vulnerabilities of location based and techniques used to attack the social mobile networks such as the social engineering, direct and indirect anonymity, eavesdropping and location cheating attacks. This is also affected by other factors such as the various mobile social-based networks themselves, and the mobile technologies platform under which each theses social network run on.

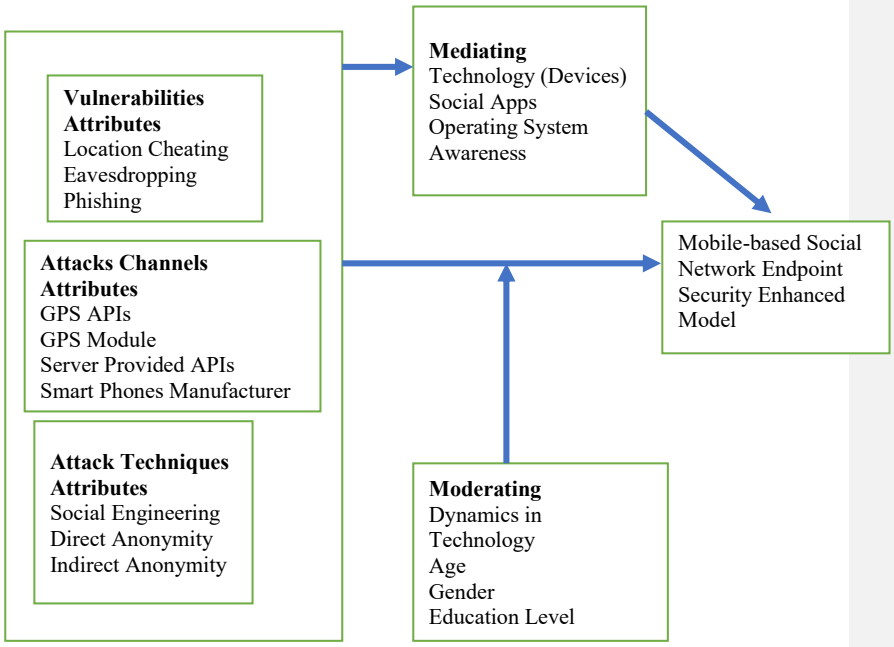


Figure 2.3 Conceptual Framework Author, 2025

2.8 Summary

Mobile social Networking is a fast-growing way of communication, more so it is an easier way of socializing among the peers. There are several of these kinds of applications some are location based while others are application based on the normal computers, but have been tailored to suit the dynamism of the ever-growing technology of the mobile phones. Currently the integration of the social networks and mobile phones are already a driving for the dissemination of information across the county, regional, national and global at large. With all these with standing, there are lots of information within each user profiles that could be a danger for the

ever-growing security concerns, which leads to use of major techniques in order to obtain the information illegally. Some of the security threats are specific to the existing social networks.

Not unless the very threats that pose to use or acquiring of an ID in any of the social networking into the mobile phones not to have been addressed then there is a danger they pose in the users' privacy among others. Blowfish, a 64bit block cipher, is an excellent choice for encryption, since it is lightweight, public domain, and highly secure even after extensive analysis.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Overview

This chapter presents the research design, the study flow, data collection and analysis by use of the research onion design (which outlines research philosophy, research approach, Research strategy, choice of research method, time horizon, data collection techniques and Techniques, and analysis). The methodology in this study, is demonstrating how the specified problems were solved by the chosen design, why it is preferred rather than other alternatives and how the empirical work was carried out.

3.2 Research Design

Churchill and Gilbert, 1979 proclaim that research design provides direction for collecting and analyzing data in a given study. Nachmias and Nachmias (2008) view research design as deriving its importance from its role as a critical thread between the theories and arguments that informs the research and the empirical data collected. According to Cooper and Schindler (2006), a research study design can be of exploratory, descriptive or explanatory type. They can overlap depending on the purpose of the research or on the stage of the research process. This study adopted all the three types in a sequential manner, starting with exploratory, from which the findings led to a descriptive design. The output of the descriptive design then informed the explanatory design, which was achieved through quasi-experiment and the development of a solution/technique to augment the security vulnerabilities identified by the quasi-experiment process. The table 3.1 gives a summary of how the four objectives were addressed in terms of methods, data collection Techniques and data analysis Techniques.

Table 3.1 A summary of the methodology and Techniques for each of the four objectives

	Objectives	**Method**	**Data Collection Technique**	**Analytical Technique**
1	To establish the challenges in today's mobile-based social networks	Mixed Method	Qualitative and quantitative (questionnaire, quasi-experimental and literature review	Thematic, SPSS, MobSF, GitHub Suite
2	To determine techniques used by adversaries to attack mobile-based social networks	Mixed method	Qualitative and quantitative (questionnaire, quasi-experimental and literature review	Thematic, SPSS, MobSF, GitHub Suite
3	To establish mechanisms used to mitigate the security challenges in mobile-based social networks	Mixed method	VAPT	MobSF & GitHub Suite
4	To develop a framework/technique to augment a mobile-based social networking security mechanism	Mixed method		Based on the findings from the literature review, surveys, and simulations, developed a proposed security technique/solution for enhancing endpoint security in mobile-based social networks to address identified vulnerabilities and incorporate best practices for mobile security, privacy, and user experience

3.3 Research Paradigms: A Methodological Reflection- The Influence of Pragmatism

The methodological approach for this thesis was guided by the pragmatist research paradigm, which prioritizes the research problem over strict adherence to a single philosophical stance (McGregor & Murnane, 2010). This worldview, as an overarching system of inquiry (TerreBlanche & Durrheim, 1999), directly justified the integration of mixed methods. The decision to combine qualitative survey insights with quantitative simulation evidence was driven by the pragmatist tenet that the most suitable methods are those that best serve the practical purpose of the investigation.

This pragmatist foundation shaped the study's core philosophical axioms. Ontologically, it supported a problem-oriented view of reality, treating endpoint security as a complex issue requiring multiple angles of attack. Epistemologically, it upheld the belief that knowledge is what is useful and actionable, necessitating both the objective, generalizable data from simulations and the contextual, experiential insights from surveys to form a complete understanding (McGregor & Murnane, 2010). This was further guided by an axiology that centrally incorporates researcher values, specifically the pursuit of a practical and effective security solution (Ponterotto, 2005).

The logic of inquiry under pragmatism was abduction, which necessitated a cyclical process of gathering and interpreting both quantitative and qualitative data. The rigor of the study was thus established through methodological triangulation, where the strengths of one method compensated for the weaknesses of the other. Ultimately, the pragmatic paradigm provided the philosophical justification for this integrated approach, framing the research problem as one that required both objective measurement and contextual understanding to develop a comprehensive and effective security model.

3.4 Research Philosophy: The Pragmatic Paradigm in Practice

The research philosophy, as the foundational layer of the research onion, represents a crucial belief about how data is collected, interpreted, and analyzed to develop knowledge (Levin, 1988). For this study, pragmatism was adopted as the guiding philosophy. As Simpson (2009) notes, pragmatism seeks to account for lived experience, and according to Harter (2007), it is concerned with the relationships between antecedents and consequences. This worldview prioritizes the research problem itself over a strict commitment to a single ontological or epistemological stance, thereby offering a suitable framework for integrating both quantitative and qualitative inquiry (Saunders & Thornhill, 2012). Adopting this perspective prior to data collection was essential for guiding a theoretical view of the social world and enriching the research endeavor (McLachlan & Garcia, 2014).

This pragmatic reasoning played out directly in how conflicting insights from different data sources were reconciled. For instance, the quantitative simulation might have demonstrated high technical efficacy against threats, while the qualitative survey revealed that users found the security protocol too intrusive, threatening adoption. A purely positivist approach would have privileged the simulation data, while a purely interpretivist one might have overvalued subjective perception. Instead, guided by pragmatism's focus on practical consequences (Harter, 2007), this conflict was treated not as a failure but as the central puzzle to be solved. The researcher engaged in an abductive reasoning process, inferring that a technically perfect model is a practical failure if it is not used. The solution was to iteratively refine the model—perhaps by simplifying its interface—thereby creating a solution that was validated as both

robust and usable, truly accounting for the "lived experience" of the end-user (Simpson, 2009).

Furthermore, the pragmatic ontology, concerned with the nature of reality as a practical problem, actively influenced the design of the simulation. The simulation was not designed as a purely abstract, positivist test but was built to reflect the complex reality of mobile social networks. This was achieved by incorporating user behavior patterns identified in the preliminary qualitative survey—such as typical usage times and application preferences—as key variables within the simulation model. For example, a malware attack was simulated during periods of high user engagement identified from the survey data. This design choice ensured the quantitative evidence was not only statistically generalizable but also contextually relevant and actionable, directly fulfilling the pragmatist imperative to focus on the consequential relationships between the security model (antecedent) and its real-world performance (consequence) (Harter, 2007).

3.4.1 Pragmatism

Pragmatism research philosophy accepts concepts to be relevant only if they support action. According to Saunders, Lewis & Thornhill (2019), pragmatics “recognize that there are many different ways of interpreting the world and undertaking research, that no single point of view can ever give the entire picture and that there may be multiple realities”.

The research looked at a global perspective and arrived at a singular reality to augment the mobile based social networks, in the process there emerged that there are multiple realities in augmenting mobile based social networks. In the perspective the research deployed to using the singular and multiple realities, from the ontology point of view.

From the epistemological standing point, the study looked at how reality, or truth or how knowledge was collected or arrived at. Knowledge can be measured through scientific rules using reliable Techniques and design. The knowledge being acquired was interpretive in nature, hence it was interpreted and explained. Therefore, by so doing the knowledge was examined by use of Techniques that suited the investigation as well as scientific designs and finally interpreted for development of the technique for augmenting security in mobile based social networks at the endpoint devices. From the forgone discussion, it is clear that a pragmatic philosophy was adopted for use in the study.

3.5 Research Approaches

The research employed a sequential approach where the weighting of the deductive and inductive approaches shifted at different stages, guided by the overarching pragmatist paradigm which justified using the most appropriate method to address the research problem (Saunders et al., 2019).

The study was initially and predominantly guided by a deductive approach. This is evident in its initial focus on using the literature to "identify theories and ideas" and to "establish the challenges" in endpoint security. This extensive review led to the development of a conceptual framework, which served as a theoretical foundation to be tested. The subsequent design of the survey questionnaire was informed by this pre-existing framework, aligning with the deductive process of moving from theory to data collection (Saunders et al., 2019). In this first major phase, the deductive approach was prioritized to structure the inquiry and establish a validated starting point based on existing knowledge.

However, the weighting of the approaches shifted as the research progressed, with a strong inductive component emerging during the data analysis and solution development phase.

After collecting survey data, the research process "necessitated simulation," and the "output of the simulation helped to identify vulnerabilities." This sequence from empirical data (survey and simulation results) to the identification of specific, emergent problems (vulnerabilities) is characteristic of an inductive logic. The culmination of this inductive process was the development of a "technique/solution to the vulnerabilities" that was generated directly from the findings, rather than being solely pre-defined from the initial literature. Thus, the later stages of the research prioritized an inductive approach to build a novel, data-driven solution.

The research did not assign equal weight to both approaches simultaneously but applied them sequentially to fulfill different purposes. The deductive approach was prioritized in the early, theory-building and testing stages, while the inductive approach was prioritized in the later, exploratory and solution-oriented stages. This strategic shifting of approaches exemplifies the flexible, problem-centered methodology advocated by the pragmatist paradigm, using deduction to ground the research in established theory and induction to generate a practical and responsive outcome (Saunders et al., 2019).

3.6 Research Strategy (Study Procedures for the Collection of Quantitative Data)

This section demonstrates the research strategy, which gave the action plan used in the implementation of the research. The researcher, adopted a sequential exploratory, descriptive, and explanatory strategies to represent different approaches to investigating the research study, each with its own purpose and characteristics. Exploratory research was used to gain a general understanding of the topic. This was done to investigate if the population under study were aware or understood the topic of study; this was done through the use of a survey. The responses from the questionnaire were then descriptively analyzed and the

findings used to guide the explanatory phase of the study. The explanatory phase, which was quasi-experimental aimed to explain how the phenomenon occurred by identifying cause-and-effect relationships. Survey strategy was used to explore the underlying information and knowledge of mobile-based social networks end point security challenges. Secondly, quasi-experimental (simulation) strategy was deployed, informed by the findings of the survey strategy to highlight the vulnerabilities of the endpoint devices. From the findings of simulation (VAPT), the study then developed a technique/solution for augmenting security in mobile based social networks endpoint device. Developing a robust security solution for mobile social network endpoints involves a multi-layered approach. Key areas include enhancing authentication, securing data transmission, protecting against malware, and implementing secure coding practices. The researcher, developed an authentication solution, given that more often, data transmission challenge has been addressed through encryption.

I. Survey strategy

3.6.1 Questionnaire Development Process

The quantitative questionnaire comprised a number of sections; Section "I", captured demographic data while Section "II, III & IV" consisted of general multiple-choice questions on use of mobile endpoint devices; security & risk awareness; and risk management questions which aimed at collecting information about mobile endpoint security vulnerabilities awareness. For the purpose of questionnaire development, the researcher adopted Churchill and Iacobucci (1979), approach that comprises nine steps to develop the research questionnaire. The developed questionnaire was piloted or pretested among computer science 4th year students at Masinde Muliro University of Science and

Technology, before a full roll-out. This was to aid in capturing for correction any ambiguity and non-clear statements or issues in the questionnaire.

(Creswell, 2002), is of the view that data collection is the process of gathering information from various sources to answer research questions. Singleton and Straits (2005); Ary,et al (2006); and Dillman (2000), are of the view that there are many channels used to collect survey research data, such as post, telephone, interview, internet-based survey, email, and by-hand delivery. This study applied internet-based survey method. The researcher shared the questionnaire to a number of WhatsApp platforms and requested the members to share widely (a snowballing effect). The researcher adopted Snowball sampling, a non-probability sampling technique, because it offers several advantages, particularly when studying hard-to-reach or hidden populations. It's a cost-effective and efficient method for locating participants through referrals, enabling access to groups that might otherwise be difficult to engage. This approach also allows researchers to gather rich data from individuals within specific social networks. From these, 257 respondents were able to return their feedback. The returned filled questionnaires, were then cleaned and entered into SPSS for analysis.

3.6.2 Data collection method(s)

To be able to create confidence, and trust between the respondents and the researcher, in the questionnaire, the purpose of the study, was clearly stated. This was also accompanied by granted permission from the Directorate of Postgraduate School, Masinde Muliro University of Science and Technology, and from National Council of Science and Technology and Innovation.

II. Quasi-Experimental (Simulation) Strategy

Quasi-experimental or simulation strategies are used when true experimental or test-bed designs are not feasible or ethical, particularly when studying real-world interventions. They offer practical advantages by allowing researchers to investigate causal relationships in natural settings, enhancing external validity. Other further advantages are: i. they are conducted in natural environments, allowing researchers to observe the effects of interventions in more realistic contexts; ii. They are flexible and adaptable. The researcher, thus, adopted this strategy because of the stated advantages.

Mobile security is a crucial aspect of information security that focuses on protecting smartphones, tablets, and other mobile devices from threats and vulnerabilities. It encompasses a range of practices, technologies, and solutions designed to safeguard mobile devices and the data they contain. The researcher adopted two prominent Techniques used in mobile security; notably: Mobile Security Framework (MobSF) and Git Hub Suite.

3.7 Research Method

Saunders and Thornhill (2019); and Punch (2003) categorize research methods into quantitative, qualitative, or mixed methods. The authors, further, broadly classify the methods into mono method, mixed method and multi methods. Different aspects, such as research questions, objectives, and topic always help guide which research method is most appropriate, (Al Zefeiti and Mohamad, 2015). The study used mixed method research design which according to Creswell (2012), is a procedure for collecting, analyzing, and mixing both quantitative and qualitative research and methods in a single study to understand a research problem. For quantitative research design, Creswell (2012) views it as one in which the researcher decides what to study; asks specific, narrow questions, collects quantifiable

data from participants; analyzes these numbers using statistics; and conducts the inquiry in an unbiased, objective manner.

3.8 Techniques Used

The MobSF (Mobile Security Framework) environment figure 3.1a and figure 3.1b is a comprehensive, open-source, and automated security testing platform designed for mobile applications. It functions as an all-in-one penetration testing toolkit that supports the static and dynamic analysis of Android, iOS, and Windows mobile apps, along with their corresponding binaries and source code. The environment is typically deployed as a web application, providing a user-friendly interface where security researchers and developers can upload an application package to receive a detailed security assessment report. The report identified vulnerabilities such as insecure data storage, improper platform usage, and code-level flaws, while also including features for dynamic analysis, such as interactive instrumented testing and network traffic analysis. By consolidating these capabilities into a single, centralized environment of MobSF, it streamlines the security auditing process, making it an essential tool for integrating security checks into the DevOps (CI/CD) pipeline and for performing efficient mobile application security assessments.

The following Techniques in table 3.2, were used in the security analysis process for the three applications:

Table 3.2: Techniques used for Security Analysis Process for the Applications

Technique	Platform	Role in Analysis
MobSF v.3.4.1	GitHub v3.4.1	Static/dynamic analysis of APKs, permission/certificate review
apkTechnique	GitHub v3.4.1	Decompiling APKs to access manifest and resources
jadx	GitHub v3.4.1	Decompiling. dex to Java source
Androguard	GitHub v3.4.1	Parsing permissions, method invocation tracing
VirusTotal	Web 5.3.0	Malware and IP/domain reputation checking
Wireshark	App	Capturing and analyzing traffic (TCP/UDP)
mitmproxy	GitHub v3.4.1	SSL inspection and traffic manipulation
ExifTechnique	GitHub v3.4.1	Reading metadata from assets
SSDeep / TLSH	GitHub v3.4.1	Binary similarity analysis
JA3 (TLS fingerprinting)	GitHub v3.4.1	Profiling app behavior via SSL signatures
KeyTechnique	CLI	Extracting certificate and signing metadata
Drozer	GitHub v3.4.1	Dynamic exploitation of exposed components
AXMLPrinter2 / axmldec	GitHub v3.4.1	Decoding manifest and permission files

Vulnerability Assessment (VA):

This entailed the application of automated Techniques as well as manual Techniques to scan the systems that are known to have security weaknesses; systems, applications and networks. The aim was to identify the security vulnerabilities, which may be misused by malicious players, so that the organizations or individuals can mitigate them before they can be used to perpetrate attacks; MobSF v3.4.1 was employed. It established possible entry points that may be used by attackers.

Penetration Testing (PT):

This was a simulated real-world attack, such as hacking attempts, to assess how well security controls and defenses actually worked in practice, specifically, GitHub v3.4.1 Techniques

were used. This involved exploiting vulnerabilities found during the VA stage in a controlled environment to evaluate the impact of security flaws.

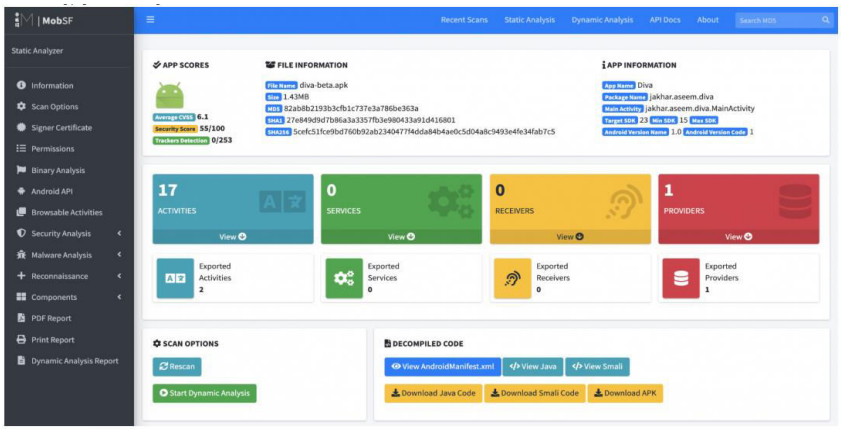


Figure 3.1a: MobSF v3.4.1- Mobile Security Framework Simulation (Source: Author)

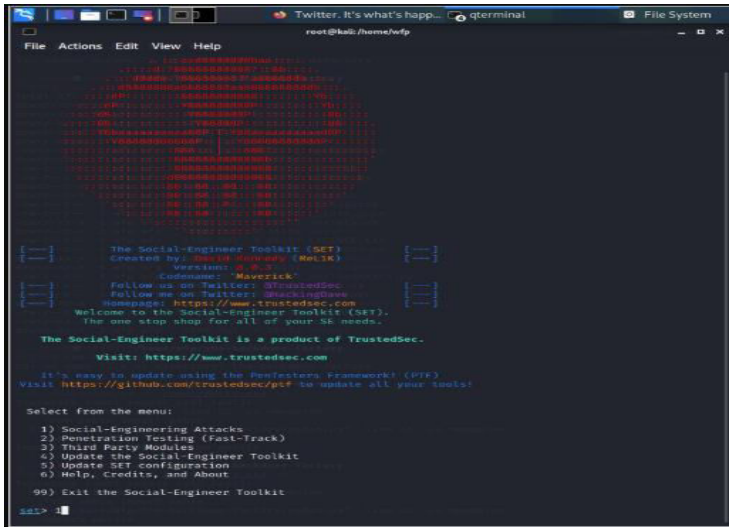


Figure 3.1b: MobSF v3.4.1- Mobile Security Framework Simulation (Source: Author)

3.8.1 Quasi-Experimental setup [Simulation]

I. Mobile Security Framework (MobSF)

MobSF is an automated, all-in-one mobile application security testing framework capable of performing static analysis, dynamic analysis, and web API 5.3.0 testing. It is open-source and supports both Android and iOS applications.

The study used this Technique to do static analysis, dynamic analysis, web API 5.3.0 testing, thereafter integrated the results with the GitHub output and also generated a report.

Static Analysis: MobSF scanned the source code of three mobile apps (WhatsApp, Facebook and X) to identify their security vulnerabilities, insecure code practices, and privacy-related issues.

Dynamic Analysis: It ran the applications in a controlled environment to monitor their behavior and identify security weaknesses during runtime.

Web API 5.3.0 Testing: MobSF v3.4 was also used to assess the security of the web APIs as they are used by mobile applications.

Easy Integration: The study, exploited this feature of the Technique, to make it possible to integrate the vulnerability assessment (VA) and penetration testing (PT), which was done using GitHub Techniques, since it is designed for easy integration with other security Techniques and continuous integration/continuous deployment (CI/CD) pipelines.

Report Generation: This feature of the Technique, made it possible to provide detailed and comprehensive reports highlighting security flaws and recommendations for remediation.

GitHub Suite

Git Hub Suite is a popular Technique for web application security testing. While primarily designed for web applications, it is also highly effective in testing the security of mobile

applications, especially those that interact with web services. Specifically, these Techniques were used to achieve penetration testing. Its main features are presented here below.

Interception Proxy: The study took advantage of this feature to allow interception and modification of HTTP/HTTPS traffic between the mobile app and its backend server.

Scanner: The study further exploited this feature by doing an Automated scanning for identifying vulnerabilities.

Intruder: The study was able to take advantage of this feature to do a customized attacks to test for vulnerabilities.

Repeater: The study, exploited this feature to facilitate manual testing by allowing the resending of requests with modified inputs.

Decoder and Comparer: Techniques for decoding data formats and comparing responses.

The study, used the Techniques:

i. to intercept and analyze traffic between mobile apps and backend services;
ii. identify vulnerabilities in mobile app web interfaces.
iii. Manually and automated security testing of mobile applications.

II. Integration of MobSF and GitHub Suite

Integrating MobSF and GitHub Suite provided a comprehensive approach to mobile security. MobSF's capabilities in static and dynamic analysis complemented GitHub Suite’s strengths in intercepting and analyzing web traffic. This combination allowed for thorough examination of both the client-side and server-side components of mobile applications.

Workflow

Initial Assessment: The study used MobSF for an initial static and dynamic analysis of the mobile app.

Traffic Analysis: The study utilized GitHub Suite's interception proxy to analyze and manipulate the traffic between the app and its backend.

Vulnerability Identification: The study employed GitHub Suite's scanning and intrusion Techniques to identify and exploit web-based vulnerabilities.

Final Report: The researcher, then combined the findings from both Techniques for a comprehensive security assessment report.

III. Development of a solution/technique

The study purposed to develop an end-to-end encryption authentication system guided by the findings from VAPT (MobSF & GitHub Suite), from the quasi-experiment; where all messages, whether text, media (like images or videos), or voice messages, are encrypted on the sender's device and can only be decrypted on the recipient's device. Developing a robust security solution for mobile social network endpoints involved a multi-layered approach. Key areas included enhancing authentication (which was the key focus of the study), securing data transmission, protecting against malware, and implementing secure coding practices. The methodology encompassed threat modeling, security requirements gathering, architectural design, implementation with security best practices, and rigorous testing. The developed system has the following characteristics:

i. Unique Key Pairs:

Each system user has a unique set of public and private keys based on Elliptic Curve Cryptography.

ii. Encryption in Transit:

When a user sends a message, it's encrypted using the recipient's public key on the sender's device. This encrypted message is then sent to the system servers and then to the recipient's device.

iii. Decryption on Recipient's Device:

Only the recipient's device, possessing the corresponding private key, can decrypt the message and make it readable.

iv. Perfect Forward Secrecy:

The system also implemented Perfect Forward Secrecy (PFS), which means that even if a private key were compromised, it wouldn't compromise past conversations.

v. Group Calls:

The system also supports end-to-end encrypted group calls, where the encryption ensures that only the participants in the call can hear or see the content.

This end-to-end encryption is a fundamental security feature of the system, protecting user privacy and communication from eavesdropping

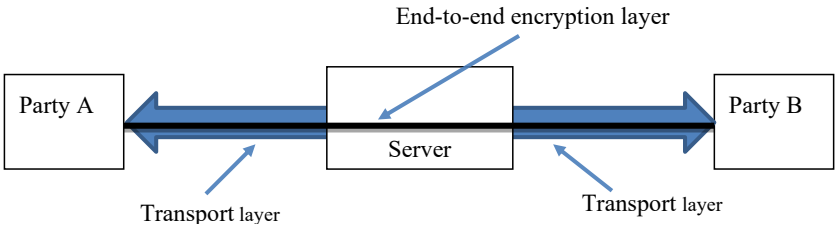


Figure 3.2: End-to-end encryption layer (Source: Author)

Threat Modeling and Security Requirements: [Based on the results from MobSF & GitHub Suite]

Identified potential threats:

Analyzed common attack vectors like phishing, malware, man-in-the-middle attacks, and data breaches. Considered both internal and external threats.

Defined security requirements:

Based on the threat model, established specific security goals for the mobile application, such as strong authentication, secure data storage, and protection against unauthorized access.

Prioritized requirements:

Determined which security features are most critical and allocated resources accordingly.

Architectural Design:

Secure Authentication:

Implemented strong authentication mechanisms beyond simple passwords, such as biometrics (fingerprint, facial recognition), multi-factor authentication (MFA), or device-bound passkeys.

Secure Data Storage:

Encrypted sensitive data both at rest (on the device) and in transit (when communicating with the server).

Secure Communication:

Used HTTPS for all network communication to prevent eavesdropping and man-in-the-middle attacks.

Secure Coding Practices:

Followed secure coding guidelines to prevent common vulnerabilities like buffer overflows, SQL injection, and cross-site scripting (XSS).

Consider a layered approach:

Implemented multiple security layers so that if one is compromised, others can still provide protection.

Implementation:

Used secure libraries and frameworks:

Leveraged well-established and secure libraries for cryptography, networking, and other security-related functionalities.

Implement input validation and sanitization:

The researcher purposed to prevent attacks by validating and sanitizing all user inputs to prevent malicious code injection.

Handle errors and exceptions securely:

The researcher implemented a proper logging and error handling mechanisms in order to avoid exposing sensitive information through error messages.

Testing:

Unit testing: Test individual components for security vulnerabilities.

Integration testing: Test how different components interact with each other from a security perspective.

Penetration testing: Simulate real-world attacks to identify weaknesses in the application.

3.9 Data Analysis

Central to any scientific research is a question that the research is trying to address. Nachmias and Nachmias (2008) have suggested that statistical techniques are a major Technique for data analysis. In this study, Statistical Package for Social Sciences (SPSS.27)

was used to analyze the collected data survey data. As the research involved different research objectives, a number of statistical techniques were used. Descriptive statistics which deal with the presentation of numerical facts, or data, in either tables or graphs form; or which summarized essential features of data- central tendency, variability, and distribution in either of these formats was used.

3.10 Validation of the MbAT

The Mobile-based Authentication Technique (MbAT) was evaluated through the deployment of a functional prototype in a controlled field environment. The testing focused on validating architectural integrity, functional correctness, and security efficacy. The results are summarized below.

Architectural Validation

Test Objective: To verify that the system's design and component interactions are sound and adhere to software engineering principles.

Method: The system's architecture was formalized and validated using standard software engineering diagrams.

Functional Validation

Test Objective: To demonstrate the prototype's operational capability in a real-world mobile environment.

Method: The installed prototype was executed on a mobile device, and key processes were documented via screen captures.

Security Validation

Test Objective: To assess the technique's effectiveness in protecting message confidentiality and ensuring access control.

Method: The security argument was validated by analyzing the system's behavior during the functional tests and verifying the implementation of its cryptographic layers.

3.11 Ethical consideration

The rule of voluntariness entails that individuals should not be forced into taking part in research. This is particularly applicable in instances where researchers had formerly depended on the 'captive audiences' as their study population, which were prisons, universities, and so on.

Intimately connected to the concept of voluntary participation is the prerequisite of informed consent. Basically, this implies that the potential research subjects should be aware of the procedures and risks associated with research and should consent to take part.

The ethical standards also demand that the researchers should not expose the participants to a scenario where they may be vulnerable to some harm due to their participation. Harm may be defined as physical and psychological. Two standards are used to assist in defending privacy of research participants.

The confidentiality of the participants is guaranteed by almost all the research- it is stated that no information that allows identifying them is going to be disclosed to anyone not directly involved in the research.

The more rigorous is the principle of anonymity that basically implies that the participant will be anonymous during the research process even to the researchers themselves. It is obviously evident that, anonymity standard is a greater assurance of privacy but in some cases it may not be achievable, particularly in cases where participants must be measured at several points in time (as in, a pre-post study).

The ethical problem concerning the right to service of a person has increasingly been a concern that researchers have had to address. No-treatment control group is frequently

necessary in good research practice - a group of participants not receiving the treatment or program under study. However, when such treatment or program can have positive outcomes, individuals in the no-treatment control group might believe that their right to equal access to services is being limited.

Despite obvious sets of ethical norms and rules, there will be instances where the necessity to conduct proper research collides with the rights of the prospective subjects. There is no code of standards that can possibly foresee all of the ethical situations. In addition, a process should be in place that will ensure that such researcher will take into account all pertinent ethical concerns in the process of developing research plans.

CHAPTER FOUR

DATA ANALYSIS AND INTERPRETATION

4.1 Overview

This chapter presents results from both the quantitative and qualitative data outlined in chapter three alongside each other according to the objectives. The study acknowledges the demographic limitations and proposes deeper, alternative interpretations of the data.

4.2 Findings from the Survey Data (Descriptive Statistics)

The study assessed the following demographic parameters as they were considered to be moderating factors of security perceptions and behaviors: Gender, Age, and level of Education. From the descriptive analysis, the results are captured in tables 4.1, 4.2 and 4.3. The first output from the analysis is a table of descriptive statistics for the variables under investigation. Analyzed broadly in terms of male and female participation or their frequency, one can conclude that there were more males at 157 (61.1%) compared to female at 100 (38.9%) table 4.1.

Table 4.1: Distribution of population by Gender

What is your Gender?

	N	%
Male	157	61.1%
Female	100	38.9%

Further, from table 4.2 it shows that of the total respondents, the age range group of 25-34 were the majority with a percentage of 137 (53.3%) followed by the age range of 35-44 at 61 (23.7%); while the lowest age range were <24 at 59 (23%).

Table 4.2: Distribution of population by Age

What is your Age Range?

	N	%
<24	59	23.0%
25- 34	137	53.3%
35- 44	61	23.7%

The study also analyzed the population distribution by the level of education. The results are captured in table 4.3. Degree holders are the majority at 143 (55.6%), followed by postgraduate at 36.6%, while from tertiary (Certificate & Diploma) had the lowest response.

Table 4.3: Distribution of population by Level of Education

What is your level of Education

	N	%
Tertiary (Certificate & Diploma)	20	7.8%
Degree	143	55.6%
Postgraduate	94	36.6%

Based on the provided demographic data, it is not possible to conclude which specific group benefits from the limitations of security frameworks; The study's sample is predominantly composed of young, highly educated males, with the largest cohort being 25-34-year-olds (53.3%) and the majority holding a degree or postgraduate qualification (92.2% combined). This demographic profile suggests that the limitations of existing security models particularly their failure to adequately protect against sophisticated social engineering, most significantly impact a digitally native, working-age population that is heavily engaged with technology. While their higher education might imply better security awareness, the inherent flaws in the frameworks (like the inability of sandboxing to prevent phishing or MTD to stop

scams on encrypted platforms) create a universal vulnerability. Therefore, the most active and connected segment of the population, as captured in this study, is simultaneously the most reliant on these technologies and the most exposed to their critical shortfalls, rather than truly "benefiting" from them.

The study's demographic composition severely limits the generalizability of its findings due to a non-representative sample. The results are skewed by a significant overrepresentation of males, young to middle-aged individuals, and highly educated respondents. Consequently, the findings primarily reflect the security perceptions of a specific, likely more technically optimistic, demographic cohort. The study's conclusions should therefore not be interpreted as representative of the broader population's behaviors or attitudes.

4.3 Mobile endpoint device, SMN and Operating System in Use

The type of mobile endpoint device significantly impacts vulnerabilities in mobile-based social networks due to variations in security features, operating systems, and user behavior. More sophisticated devices with robust security measures offer better protection, while older or less secure devices are more susceptible to attacks. From table 4.4, more than three-quarters of the respondents were using smartphones 225 (87.5%), while a meagre 32 (12.5%) used tablets.

Table 4.4: Type of mobile endpoint device used

What type of mobile endpoint device do you use?

	N	%
Smartphone	225	87.5%
Tablet	32	12.5%

Understanding the operating system (OS) of a mobile endpoint device is crucial for effective endpoint security because different OS versions have varying security vulnerabilities and features. From the table 4.5, Android users, were the majority at 211 (82.1%), while those whose phones had iOS were 46 (17.9%).

Table 4.5: A Summary of Operating System of the endpoint device

What operating system does your device use?

	N	%
iOS	46	17.9%
Android	211	82.1%

Social media's widespread use significantly impacts endpoint device security vulnerability by creating new avenues for cyberattacks and data breaches. Social engineering tactics, phishing attempts, and the spread of malware are amplified through social networking platforms, making users and their devices more susceptible to compromise. The interconnected nature of social media also exposes users' personal information and relationships, which can be leveraged by attackers to orchestrate more sophisticated attacks against individuals and organizations. From table 4.6, it can be seen that a greater percentage of the respondents 177 (68.9%) were using general purpose messaging apps and media-sharing sites; 47 (18.3%) used messaging apps: WhatsApp; while 33 (12.8%) of the respondents used microblogging sites and specialized platforms.

Table 4.6: Type of Social Networking Sites Regularly Used

What social networking sites do you regularly use?

	N	%
messaging apps: WhatsApp	47	18.3%
general-purpose; messaging apps and media-sharing sites	177	68.9%
microblogging sites and specialized platforms	33	12.8%

4.4 Mode of Security in Use at the EndPoint

Using a strong password or passcode on one's mobile device is crucial for protecting personal information from notably, unauthorized access, financial fraud, and identity theft. A strong password makes it significantly more difficult for hackers to gain access to a device which is not theirs and the sensitive data contained in it. From table 4.7 the respondents, majority, 112 (43.6%) used passwords, but were not sure of the strength, 96 (37.4%); while 49 (19.1%) did not use any password.

Table 4.7: Use of Strong Password or Passcode on Endpoint Mobile Device

Do you use a strong password or passcode on your mobile device?

	N	%
Yes	96	37.4%
I use password, but I'm not sure whether it is strong	112	43.6%
No, I don't use any password	49	19.1%

Multi-factor authentication (MFA) is crucial for securing social media accounts because it adds a second layer of security beyond just a password, significantly reducing the risk of unauthorized access. Even if a password is stolen or compromised, an attacker would still need the second factor (like a code from your phone) to log in. This makes it much harder for cybercriminals to hack into accounts and steal personal information or cause other malicious damage. From table 4.8, 85 (33.1%), indicated that they did not use MFA, but were planning to activate the same; 65 (25.3%) on the other hand acknowledged that they used MFA on some social networking sites; 60 (23.3%) were not using MFA; while 47 (18.3%) were using MFA in all the social networking sites.

Table 4.8: Use of Multi-Factor Authentication on the Social Networking Accounts

Do you use multi-factor authentication on your social networking accounts?

	N	%
Yes, I use MFA on all of my social media accounts	47	18.3%
Yes, I use MFA on some of my social media accounts	65	25.3%
No, but I plan to enable MFA	85	33.1%
No, and I don't plan to use MFA	60	23.3%

Updating one's device's software or operating system (OS) is crucial for maintaining the security and performance of the social networking accounts. Regular updates patches vulnerabilities that hackers can exploit to access the data stored in the device, and they also improve overall device performance and introduce new features. From table 4.9, 107 (41.6%) who were the majority indicated they updated the device's software or OS for bug fixing, new features, performance improvements and security enhancements; 45 (17.5%)

updated for new features; 42 (16.3%) updated for performance improvements; 35 (13.6%) updated for security enhancements, 27 (10.5%) never updated their software or OS, while 1 (0.4%) updated for bug fixing.

Table 4.9: Update of Software or OS

Have you ever updated your device's software or operating system?

	N	%
For bug fixes	1	0.4%
For new features	45	17.5%
For performance improvements	42	16.3%
For security enhancements	35	13.6%
All of the above	107	41.6%
None of the above	27	10.5%

Using antivirus or security software on one's device is crucial for protecting the social networking accounts from cyber threats. It safeguards the accounts from malware, phishing attacks, and other malicious activities that can compromise personal information and online presence. From the table 4.10, 203 (79.0%) regularly used antivirus or security software; while 54 (21.0%) did not use the antivirus or security software.

Table 4:10: Regular Use of Antivirus or Security Software on the device

Table 4:10: Regular Use of Antivirus or Security Software on the device

Do you regularly use antivirus or security software on your device?

	N	%
Yes, I regularly use antivirus/security software	203	79.0%
No, I do not use antivirus/security software	54	21.0%

Adjusting privacy settings after a breach is crucial for regaining control over one's social media accounts and minimizing further risk. It helps limit who can see device information, reducing the potential for misuse and identity theft. By reviewing and customizing settings, one can ensure only trusted connections have access and prevent unauthorized access to the profile and posts. From table 4.11, 145 (56.4%), more than half of the respondents had not adjusted their privacy settings, 41 (16.0%) acknowledged having adjusted their privacy settings by limiting activity tracking; 26 (10.1%) indicated that they adjusted their privacy settings by managing advertising preferences; 25 (9.7%) on the other hand indicated the adjusted their privacy settings by deleting cookies; 20 (7.8%) indicated that they adjusted their privacy settings by using strong password and two factor authentication.

Table 4.11: Adjusting Privacy Settings after a Breach

Have you ever adjusted your privacy settings after a breach?

	N	%
By limit activity tracking	41	16.0%
By deleting cookies	25	9.7%
By managing advertising preferences,	26	10.1%
By using strong passwords and two-factor authentication	20	7.8%
None of the above	145	56.4%

Using a VPN on one's mobile device enhances the security and privacy of the social media activity. By encrypting the internet traffic and masking the IP address, a VPN makes it more difficult for others to track one's online activity and access personal information, especially when using public Wi-Fi. From the table 4.12, a majority of 218 (84.8%) indicated that they don't use VPN, while a meagre 39 (15.2%) indicated that they do use VPN.

Table 4.12: Use of VPN on the Endpoint Device

Do you use virtual private networks (VPNs) on your mobile device?	N	%
Yes	39	15.2%
No	218	84.8%

Managing app permissions on the endpoint mobile device is crucial for maintaining privacy and security. Granting unnecessary or excessive permissions, especially to social networking apps, can expose sensitive data to potential misuse by developers or malicious actors. Regularly reviewing and adjusting permissions, like those for location, camera, and microphone, helps protect one's personal information from being collected, tracked, or exploited without one's knowledge. From table 4.13, 79 (30.7%) indicated that they managed permission by denying access completely; 63 (24.5%) achieved permission management while using the app; 59 (23.0%) indicated that they asked every time and 56 (21.0%) indicated that they asked all the time.

Table 4.13: Managing Permission for mobile apps, allowing access to specific features

Do you often manage permissions for your mobile apps e.g. allowing an app access to a specific feature (like location, camera, or microphone)?

	N	%
All the time	56	21.8%
While using the app	63	24.5%
Asking every time	59	23.0%
Denying access completely	79	30.7%

Data sharing practices among mobile apps are significant due to their potential impact on user privacy and security. Understanding these practices is crucial for making informed decisions about app usage and protecting personal information. This would allow users to actively manage their privacy settings, opt out of certain data collection practices, and potentially choose more privacy-respecting apps. From the table 4.14, the results shows that 76 (29.6%), were aware of privacy settings; 69 (26.8%) were considering use of Techniques like VPN to monitor network traffic; 67 (26.1%) were however not aware; while 45 (17.5%) were reviewing app permissions.

Table 4.14: Awareness of data sharing practices among mobile apps

Are you aware of data sharing practices among mobile apps?

	N	%
reviewing app permissions	45	17.5%
privacy settings	76	29.6%
considering using privacy tools like VPNs to monitor network traffic	69	26.8%
No I am not aware	67	26.1%

Awareness of potential security risks on social networking sites is crucial for individuals and organizations to protect personal information, brand reputation, and financial assets. Understanding these risks allows for proactive mitigation strategies, preventing cyberattacks, data breaches, and other malicious activities. From the table 4.15, 88 (34.2%) were aware of all the options of potential security risks associated with social networking sites given; 59 (23.0%) were aware of phishing as a potential security risk associated with social networking sites; 57 (22.2%) were aware of scams; 38 (14.8%) were aware of click-jacking; 8 (3.1%) were aware of phreaking; while 7 (2.7%) were aware of a few.

Table 4.15: Awareness of the potential security risks associated with social networking sites

Are you aware of the following potential security risks associated with social networking sites?	N	%
Click-jacking	38	14.8%
Phishing	59	23.0%
Phreaking	8	3.1%
Scams	57	22.2%
A few	7	2.7%
All of them	88	34.2%

Regularly checking for and installing updates and patches for mobile apps is crucial for maintaining security, performance, and functionality. Updates often include security patches that address vulnerabilities, preventing potential data breaches and malware infections. They also fix bugs, improve performance, and introduce new features, enhancing the overall user experience. From the survey results table 4.16, more than half of the respondents 131 (51.0%) have enabled automatic updates; 74 (28.8%) do manual checks; while 52 (20.2%) neither do automatic updates or manual checks.

Table 4.16: Regularly Checking for Updates and Patches for Mobile Apps

Do you regularly check for updates and patches for your mobile apps?	N	%
I have Enabled Automatic Updates	131	51.0%
I do Manual Checks	74	28.8%
None of the above	52	20.2%

Using one's mobile device on an unsecured Wi-Fi network can significantly increases the risk of exposing one's sensitive data to malicious actors. Hackers can intercept the internet traffic, steal personal information, and even install malware on these devices. This can lead to identity theft, financial loss, and other serious consequences. From the table 4.17, 103 (40.1%), indicated a no- they do not use their mobile device on unsecured Wi-Fi networks; 72 (28.0%) returned yes- indicating that they use their mobile devices on unsecured Wi-Fi without any precaution; 57 (22.2%) indicated they use VPN; while 25 (9.7%) indicated that they turned their phones into a hotspot.

Table 4.17: Use of mobile device on unsecured Wi-Fi networks

Do you use your mobile device on unsecured Wi-Fi networks?

	N	%
Yes without any precaution	72	28.0%
I Use a VPN	57	22.2%
I Turn my phone into a hotspot	25	9.7%
No	103	40.1%

Having a social networking account compromised on a mobile device can have significant consequences, including unauthorized access to personal information, potential financial loss, reputational damage, and the spread of malware or scams. Hackers can use compromised accounts to steal sensitive data, impersonate the user, or even spread malicious software to the user's contacts. From the respondents, table 4.18; 170 (66.1%), about two-thirds indicated that they have not been hacked; while the remaining one-third, 87 (33.9%) acknowledged having been hacked.

Table 4.18: Having been hacked or having the account compromised while using social networking sites on mobile device.

Have you ever been hacked or had your account compromised while using social networking sites on your mobile device?	N	%
Yes	87	33.9%
No	170	66.1%

Receiving suspicious emails, messages, or links while using social networking sites is a significant warning sign of potential scams and cyberattacks. These messages are often attempts to trick you into revealing personal information, downloading malware, or unknowingly participating in fraudulent activities. From the table 4.19, 144 (56%) indicated they had not received any suspicious emails, while 113 (44%) had received suspicious emails.

Table 4.19: Receiving suspicious emails, messages, or links on your mobile device while using social networking sites.

Have you ever received suspicious emails, messages, or links on your mobile device while using social networking sites?	N	%
Yes	113	44.0%
No	144	56.0%

Familiarity with device privacy settings is crucial for controlling how your personal information is handled online and offline. Understanding and adjusting these settings can significantly reduce the risk of data breaches, identity theft, and unwanted tracking, while also empowering you to decide what information you share with others. From the table 4.20, the results show that139 (54.1%) were very familiar, while 118 (45.9%) were moderately familiar.

Table 4.20: Familiarity with privacy settings on mobile device

How familiar are you with privacy settings on your device?

	N	%
Moderately familiar	118	45.9%
Very familiar	139	54.1%

Cross-tabulating app permissions with demographics provides valuable insights into how different user groups interact with and perceive app access to features like location, camera, or microphone. This analysis reveals the patterns in permission preferences based on age, gender, education, or other demographic factors, allowing developers to tailor permission requests and improve user experience and privacy. The table 4.21, captures the responses on the question, on how often one manages permissions for his/her mobile apps e.g. allowing an app access to a specific feature (like location, camera, or microphone) cross tabulated with gender. From this, a total of 79 (30.7%) responded to denying access completely; from which 51 (32.5%) were males and 28 (28.0%) were females; 63 (24.5%) of the total, indicated while using the app, from this males were 19 (12.1%), while females were 44 (44.0%); 59 (23.0%) of the total response, indicated that they were asking every time, from

this 50 (31.8%) were males and 9 (9%) were females; the remaining 56 (21.8%) of the total, indicated all the time from which 37 (23.6%) were males and 19 (19.0%) were females.

Table 4.21: Crosstabulation on how often one manages permissions for his/her mobile apps such as allowing an app access to a specific feature (like location, camera, or microphone) and Gender

Do you often manage permissions for your mobile apps e.g. allowing an app access to a specific feature (like location, camera, or microphone)? * What is your Gender? Crosstabulation

		What is your Gender?					
		Male		Female		Total	
		N	%	N	%	N	%
Do you often manage permissions for your mobile apps e.g. allowing an app access to a specific feature (like location, camera, or microphone)?	All the time	37	23.6%	19	19.0%	56	21.8%
	While using the app	19	12.1%	44	44.0%	63	24.5%
	Asking every time	50	31.8%	9	9.0%	59	23.0%
	Denying access completely	51	32.5%	28	28.0%	79	30.7%
Total		157	100.0%	100	100.0%	257	100.0%

Cross tabulating gender and experiences of security incidents/data breaches on social media via mobile devices can reveal potential disparities in how these events affect different genders. The researcher thus, did this analysis with a mind of informing targeted interventions and security awareness campaigns to mitigate risks for all users. It was to help identify if certain genders are more susceptible to specific types of attacks or if they experience these incidents differently. From the findings table 4.22, males are more susceptible to attacks at 80 (51.0%) than females at 35 (35.0%).

Table 4.22: Has experienced any security incidents or data breaches while using social networking sites on mobile device.

Have you ever experienced any security incidents or data breaches while using social networking sites on your mobile device? * What is your Gender? Crosstabulation

		What is your Gender?					
		Male		Female		Total	
		N	%	N	%	N	%
Have you ever experienced any security incidents or data breaches while using social networking sites on your mobile device?	Yes	80	51.0%	35	35.0%	115	44.7%
	No	77	49.0%	65	65.0%	142	55.3%
Total		157	100.0%	100	100.0%	257	100.0%

Correlation between mobile threats and mobile social network

To find out whether there is correlation between mobile threats and mobile based social network, correlation analysis was carried out. Pearson's correlation co-efficient was used to determine the degree of relationship between mobile threat and mobile based social network. The coefficient ranges between -1 to +1. The closer the correlation coefficient was to -1 or +1, the greater the degree of relationship between the two variables. When the coefficient equals to 0, there is no systematic relationship between the two variables.

The connection between mobile threats and mobile social networks is significant because social networks have become a prime target for cybercriminals, and mobile devices are the primary way users access these networks. This creates a dangerous convergence where malicious actors can exploit both the platforms and the devices to launch attacks, steal data, and compromise user security. The analysis from Table 4.23 shows that there is no significant relationship between the mobile threats and the mobile based social network. All the correlation are way far from the -1 or +1 across the threats.

Table 4.23: Correlation between mobile threats and mobile social network

	Virus, Trojan horses	identity theft	infected files from external forces	internal abuse	denial of service	bots and botnets	organized criminal attack	
Pearson Correlation	mobile based social networking application used in the phone	.014	.027	.008	-.048	-.089	.006	.118

Linear Regression (Model Summary)

To find out how threats are vulnerable to today's mobile based social network. Linear regression was used to test whether there were any significant threats that was vulnerable in today's mobile based social network. Similarly, a t-test analysis was also used to find out whether there was any significant difference in threats that are vulnerable to today's mobile based social network. The null hypothesis was tested at 0.05 significance level. From the analysis Table 4.24, R2 is 0.022 or 2% which imply that the variation of threats to mobile based social network is weak.

Table 4.24: Linear Regression (Model Summary)

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.150a	.022	.008	.863
a. Dependent Variable: mobile based social networking application used in the phone				

b. Predictors: (Constant), organized criminal attack, identity theft, internal abuse, infected files from external forces, bots and botnets, Virus, Trojan horses, denial of service

To find out the nature of the attack technique on mobile based social network, a linear regression test was done and the results are captured in table 4.25. The null hypothesis was tested at 0.05 significance level. From the analysis, R2 is 0.014 or 1.4%, which imply that the variation of the attack techniques to mobile based social network is very weak and that the model does not fit the data well.

Table 4.25: Linear Regression (Model Summary)

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.118a	.014	.002	.866
a. Dependent Variable: mobile based social networking application used in the phone				
b. Predictors: (Constant), phishing, direct anonymity, eavesdropping, social engineering, location cheating attacks, indirect anonymity				

From the analysis of Linear Regression (ANOVA) Table 4.26, the F value of 1.566 is greater than the significance value of 0.05, therefore the hypothesis is accepted that there is variation between the mobile threats and mobile based social network and that the threats cannot explain the vulnerabilities in mobile based social network.

Table 4.26: Linear Regression (ANOVA)

Model		Sum of Squares	df	Mean Square	F	Sig.	
1	Regression	8.158	7	1.165	1.566	.143b	
	Residual	355.018	477	.744			
	Total	363.175	484				
a. Dependent Variable: mobile based social networking application used in the phone							
b. Predictors: (Constant), organized criminal attack, identity theft, internal abuse, infected files from external forces, bots and botnets, Virus, Trojan horses, denial of service							

Linear regression (Coefficients)

Linear regression, through its coefficients, was used to test or analyze the significance of the relationship between the various factors related to mobile-based social networking applications and potential cyber threats. The coefficient table 4.27 was used to determine whether mobile threats and mobile based social network are significantly related and the strength of their relationship. From table 4.27, the t values help in determining the relative importance of each threat in the model. Since all the significance values are greater than 0.05 the hypothesis was accepted that there is variation between the mobile threats and mobile based social network and that the threats cannot explain the vulnerabilities in mobile-based social network.

Table 4.27: Linear regression (Coefficients)

Model	Unstandardized Coefficients	Standardized Coefficients	T			
	B	Std. Error	Beta	Sig.		
1	(Constant)	1.850	.166		11.139	.000
	Virus, Trojan horses	-.003	.045	-.003	-.061	.951
	identity theft	.044	.044	.048	.985	.325
	infected files from external forces	.011	.048	.012	.231	.817
	internal abuse	-.007	.045	-.008	-.155	.877
	denial of service	-.097	.047	-.116	-2.076	.038
	bots and botnets	.025	.046	.030	.552	.581
	organized criminal attack	.090	.040	.102	2.224	.027
a. Dependent Variable: mobile based social networking application used in the phone						

4.5 Techniques Used by Adversaries to Attack MSN

To establish techniques used by adversaries to attack mobile-based social networks

The study sought to find out the techniques that are normally used to attack mobile-based social network. A Likert scale was prepared in which the respondents were to answer 1-Very Often, 2-Often, 3-Fairely often, 4-Rarely 5- never experienced any attack. The result of the Likert scale is shown in table 4.28.

From the Likert scale, majority of the respondents 62 (24%) agreed that the mobile social networks are often attacked using social engineering technique while 75 (29%) of the

respondents agreed that the mobile social networks are fairly often attacked using direct anonymity. On the other hand, 75 (29%) of the respondents agreed that the mobile social networks are rarely attacked using indirect anonymity. The Likert also shows that 67 (26%) of the respondents agreeing that eaves dropping rarely often used to attack mobile social network while 70 (27%) of the respondents agreed that location cheating attacks is often used to attack mobile social network. The Likert also showed that 67 (26%) of the respondents agreed that phishing technique is often used to attack mobile social network.

Table 4.28: Likert scale on techniques used to attack mobile based social network

Serial No.	Item	1	2	3	4	5
1	Social engineering	44	62	54	59	38
		(17%)	(24%)	(21%)	(23%)	(15%)
2	Direct anonymity	33	51	75	59	39
		(13%)	(20%)	(29%)	(23%)	(15%)
3	Indirect anonymity	26	56	56	75	44
		(10%)	(22%)	(22%)	(29%)	(17%)
4	Eavesdropping	49	46	56	67	39
		(19%)	(18%)	(22%)	(26%)	(15%)
5	Location cheating attacks	33	70	62	46	46
		(13%)	(27%)	(24%)	(18%)	(18%)
6	Phishing	33	67	64	52	41
		(13%)	(26%)	(25%)	(20%)	(16%)

The high reporting of perceptible attacks like social engineering (24% "often") and phishing (26% "often") may reflect the experiences of the sample's predominantly young, educated, and male cohort, who are frequent users of mobile social networks and perhaps more attuned to these specific threats. Conversely, the lower perceived prevalence of stealthier techniques

like eavesdropping (26% "rarely") could indicate either their genuine infrequency or, more likely, the difficulty for average users to detect such covert attacks. Therefore, the results should be interpreted not as an objective measure of attack frequency, but as a map of the *perceived* threat landscape from a specific user group, where visible, interaction-based scams are more readily identified than sophisticated, background technical exploits.

Based on the survey findings, attackers prioritize techniques that exploit the intersection of platform vulnerabilities and predictable user behavior, following a path of least resistance. The most appealing attacks fall into three categories:

Permission Exploitation: Leveraging dangerously broad but legitimately granted permissions (like microphone or camera access), which users often approve without scrutiny, to enable direct data leakage.

Architectural Flaws: Exploiting silent and reliable platform vulnerabilities, such as buffer overflows or misconfigured services, which require no user interaction and offer scalable, predictable success.

UI Manipulation: Using techniques like overlay attacks to weaponize user trust, manipulating individuals into surrendering credentials by presenting fake but convincing interfaces over genuine apps.

In essence, attackers strategically target weaknesses where technical flaws are enabled by common user habits, such as permission fatigue and overconfidence.

4.6 Challenges in Securing Mobile Social Networks

The researcher also sought to find out challenges to effective implementation of security on social mobile networks as seen from table 4.29 (1-Strongly disagree, 2-Disagree, 3-Neither disagree, 4-Agree, 5-Strongly Agree).

From the analysis, 72(28%) of the respondents strongly agreed that rapid evolution of information and communication technologies poses a challenge to effective implementation of security on social mobile networks while 72(28%) of the respondents strongly agree that insufficient funds is the major challenge. The analysis also showed that 77(30%) of the respondents strongly agreeing that Constant evolution of the nature of cyber threats possess a major challenge. While 85(33%) of the respondents agreed that Inadequate and incomplete protection of computer systems poses a challenge effective implementation of security on social mobile networks. The Likert scale also shows that 96(37%) of the respondents agreeing that Weak links in the global information infrastructure as the major challenge while on the other hand 85(33%) of the respondents agreed that Lack of adequate legal frameworks as the major challenge while 92(36%) of the respondents agreed that Lack of appropriate organizational structures to deal with mobile-based social networks as the major challenge. The analysis also shows that 80(31%) respondents agreed that Lack of appropriate national and global organizational structures to deal with mobile-based social networks incidents as the major challenge while 77(30%) of the respondents agreed that Lack of basic awareness among users as the challenge to effective implementation of security on social mobile networks. The analysis also showed that 77(30%) of the respondents agreed that Lack of secure software and ICT-based applications as the major challenge while 77(30%) of the respondents agreed that Lack of law enforcements to deal with mobile-based social networks challenge as the major challenge to effective implementation of security on social mobile networks.

Table 4.29: Challenges in securing MSN

NO	ITEM	1	2	3	4	5
a	Rapid evolution of information and communication technologies.	51	44	26	64	72
		(20%)	(17%)	(10%)	(25%)	(28%)
b	Insufficient funding	46	46	52	72	41
		(18%)	(18%)	(20%)	(28%)	(16%)
c	Constant evolution of the nature of cyber threats	41	41	31	67	77
		(16%)	(16%)	(12%)	(26%)	(30%)
d	Inadequate and incomplete protection of computer systems	26	51	36	87	57
		(10%)	(20%)	(14%)	(34%)	(22%)
e	Weak links in the global information infrastructure	23	51	51	96	36
		(9%)	(20%)	(20%)	(37%)	(14%)
f	Lack of adequate legal frameworks	33	44	49	85	46
		(13%)	(17%)	(19%)	(33%)	(18%)
g	Lack of appropriate organizational structures to deal with mobile-based social networks incidents	21	41	49	92	54
		(8%)	(16%)	(19%)	(36%)	(21%)
h	Lack of appropriate national and global organizational structures to deal with mobile-based social networks incidents	23	51	41	80	62
		(9%)	(20%)	(16%)	(31%)	(24%)
i	Lack of basic awareness among users	44	31	31	77	74
		(17%)	(12%)	(12%)	(30%)	(29%)
j	Lack of secure software and ICT-based applications	41	56	44	77	39
		(16%)	(22%)	(17%)	(30%)	(15%)
k	Lack of law enforcements to deal with mobile-based social networks challenge.	39	44	33	77	64
		(15%)	(17%)	(13%)	(30%)	(25%)

The results highlight that the challenges are not just technical but are deeply rooted in human, organizational, and legal factors.

Human Factor: Lack of user awareness is a critical vulnerability, often considered the weakest link in security.

Pace of Change: The rapid evolution of both technology and cyber threats creates a constant catch-up game for security measures.

Structural Deficiencies: A lack of robust organizational structures, legal frameworks, and law enforcement capabilities specifically tailored to mobile social networks is a major hindrance.

Systemic Vulnerabilities: Concerns about weak global infrastructure and inadequate system protection point to broader, systemic security issues.

Thus, in the context of challenges in securing MSN, it can be summarized that the survey paints a picture of a multi-faceted problem where technological advancements are outpacing the human, organizational, and legal frameworks needed to ensure security, with user awareness being a particularly acute concern.

The researcher also sought to investigate whether there was correlation between mobile attacks technique and mobile based social network. Thus, correlation test analysis was carried out. Pearson's correlation co-efficient was used to determine the degree of relationship between mobile attack technique and mobile based social network. From table 2.30, the coefficient ranges between -1 to +1. The closer the correlation coefficient is to -1 or +1, the greater the degree of relatedness between the two variables. When the coefficient equals to 0, there is no systematic relationship between the two variables.

The analysis shows that there is no significant relationship between the mobile attack techniques and the mobile-based social network.

The correlation coefficients for all tested attack techniques were extremely close to zero (e.g., -.010, .014, -.045). This implied that the choice of social networking app did not predict or correlate with the likelihood of being targeted by a specific attack method.

In practical terms, this finding suggests that all users of mobile social networks are equally vulnerable to these common attack techniques, regardless of which specific application they use. The threat from phishing, social engineering, eavesdropping, among others, is universal across the mobile social media landscape, not tied to any one platform.

Table 4.30: Correlation between mobile attacks technique and mobile social network

	social engineering	direct anonymity	Indirect anonymity	Eaves dropping	location cheating attacks	phishing	
Pearson Correlation	mobile based social networking application used in the phone	-.010	.014	-.045	-.005	-.067	-.009

The survey sample was skewed towards a young, educated, male, Android-using demographic. Explicitly acknowledging this limitation prevents overgeneralization, enhances the credibility of the findings by correctly scoping their applicability, and suggests that the weak statistical correlations found might be due to the sample's homogeneity rather than a true absence of relationship.

Given the weak quantitative models, the results are better understood through alternative lenses. A systems thinking approach viewed security as a complex system where the interconnections between user behavior, device types, and app permissions are more critical than any single threat. A human-centric interpretation revealed key behavioral insights, such as the "awareness-action gap" where users installed antivirus software but neglect proactive

measures like VPNs and strong passwords, and the "primacy of usability," where convenience consistently trumps security.

By framing its findings within these demographic and interpretive contexts, the research more accurately identified the core security challenge: not isolated threats, but the complex interplay between user psychology, behavior, and technology.

4.7 Quasi-Experimental setup [Simulation] Results and Interpretation.

During the simulation tests for WhatsApp vulnerabilities, the simulators captured four different vulnerabilities with different common vulnerability scoring system (CVSS), score ratings. A comprehensive report, is however attached at the appendix I. Table 4.31, shows each of the vulnerabilities, their Common Vulnerabilities and Exposures (CVE), descriptions and CVSS score and figure 4.1 is a sample of the simulation output. The four CVSS scores, shows that the highest is 8.1, followed by 7.5, 7.0, all having high risk factor; the fourth had 5.5, rated as medium risk.

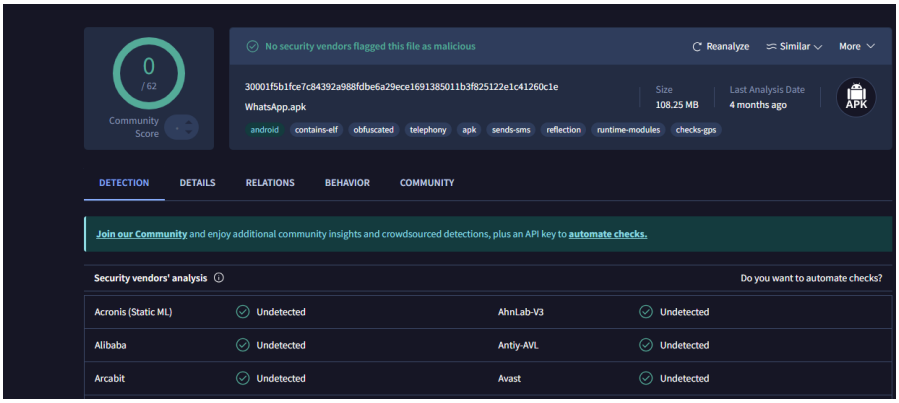


Figure 4.1: Sample Simulated output for WhatsApp (Source: Author)

Table 4.31: WhatsApp Vulnerabilities, Summary

WHATSAPP VULNERABILITIES			
Vulnerability	CVE	Description	CVSS Score
Dangerous permissions like RECORD_AUDIO, CAMERA, ACCESS_FINE_LOCATION, READ_CONTACTS, SEND_SMS can be highly exploited if accessed	CVE-2022-36934 CVE-2021-24027	Media access vulnerabilities in applications Buffer overflow in WhatsApp from video calls	8.1 (High)
Use of deprecated permissions such as GET_TASKS, INSTALL_SHORTCUT are a call for improvement in their Software Development Life Cycle management procedure.	N/A	It is generally flagged as a bad practice hence has no CVE attached to it	5.5 (Medium)
Availability of custom permissions without proper protection e.g. com.whatsapp.DYNAMIC_ RECEIVER_NOT_EXPORTED_PER MISSION	CVE-2022-20474	The CVE is connected to exported broadcast receivers without permission.	7 (High)

Directory traversal due to the identified file paths and obtained libraries	CVE-2022-1476	WhatsApp directory traversal vulnerability	7.5 (High)

Facebook was equally subjected to the same vulnerability testing. As opposed to WhatsApp vulnerabilities testing, the simulators captured two different vulnerabilities with different common vulnerability scoring system (CVSS), score ratings. Table 4.32, shows each of the vulnerabilities, their Common Vulnerabilities and Exposures (CVE), descriptions and CVSS score, while figure 4.2 show the simulated application details for Facebook. The highest risk score rating of the two was 8.0, followed by a score rating of 7.1; both indicating high risk.

Table 4.32: Facebook Vulnerabilities, Summary

FACEBOOK VULNERABILITIES			
Vulnerability	CVE	Description	CVSS Score
High risk permissions such as RECORD_AUDIO, READ_PHONE_STATE, ACCESS_FINE_ LOCATION, ACCESS_MEDIA_LOCATION, GET_ACCOUNTS	CVE-2020-1910 CVE-2023-40538	Buffer overflow in Facebook Messenger due to the handling of audio. Sensitive leakage of permissions	8.0 (High)
Custom receiver and service abuse risk due to the availability of services such as Login Notification Service and Facebook Push Server Registrar. If they are not hardened, they can provide an opportunity to attackers for privilege escalation.	CVE-2022-24706	The CVE is connected to misconfigured broadcast receivers in Facebook and the applications that are connected to it.	7.1 (High)



Basic properties

MD5	81c7fc9f4191b749abf4b1bae9b368f2
SHA-1	410f9a00a14d34d40dbb39ffc993db166e12b1e4
SHA-256	18b004102aaba568e42db3fe54b324dfc236e4b104fb9e3e5e1c16be436e7a2f
Vhash	4847d5b6a8ea6ff550681e6ae90376b2
SSDEEP	1572864:QNdNINuWX5qLEN0zVMYQkShzaWYewCLR1j2ifJ02rgPfj9OGOEu27h:tNuqqL60zVMpkizMCd902rWrgvSh
TLSH	T1A70812F7E7419C3ADE77603281205532DD2E8E299E13578B2409722EADB37D58F887C6
Permhash	f97fb62417f01ab011170d23a2f0bb44c07eed8104fb052e38eece8ee6188bd1
File type	Android executable mobile android apk
Magic	Zip archive data, at least v0.0 to extract, compression method=deflate
TrID	Android Package (44.2%) | Java Archive (22.1%) | Sweet Home 3D Design (generic) (17.2%) | Windows Media Player skin (9.8%) | ZIP compres…
File size	76.79 MB (80521801 bytes)

History

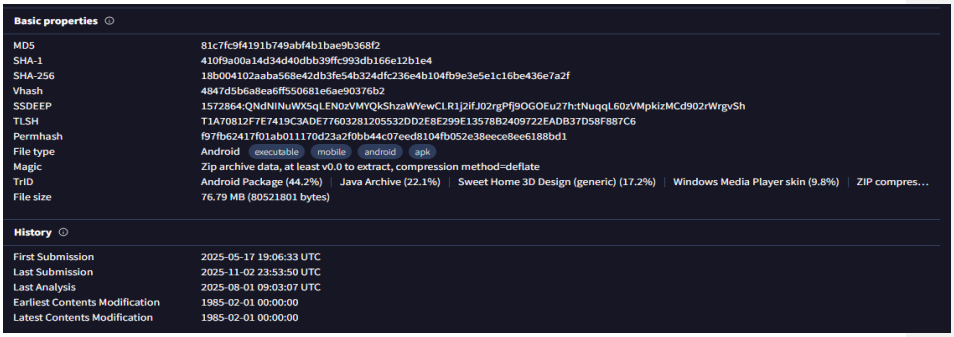
First Submission	2025-05-17 19:06:33 UTC
Last Submission	2025-11-02 23:53:50 UTC
Last Analysis	2025-08-01 09:03:07 UTC
Earliest Contents Modification	1985-02-01 00:00:00
Latest Contents Modification	1985-02-01 00:00:00

Figure 4.2: Simulated application Details for Facebook (Source: Author)

For X vulnerability testing, the same simulation procedures were undertaken. Five output results were gotten, with 3 different high CVSS score ratings of 7.8, 7.0 and 7.0; and two medium CVSS score ratings of 5.5 and 5.0. The results are captured in table 4.33, while figure 4.3 shows a sample permisions for X. The CVSS scores indicate that the simulation has vulnerabilities with varying degrees of severity. Three vulnerabilities with scores of 7.8, 7.0, and 7.0 are considered high severity, requiring prompt attention and remediation. Two vulnerabilities with scores of 5.5 and 5.0 are of medium severity, needing attention within a reasonable timeframe. The vulnerabilities with scores of 7.8, 7.0, and 7.0 fall into high-risk category. This means they pose a significant threat and could allow unauthorized access or control over the system. They require prompt attention and mitigation strategies like patching or implementing countermeasures.

Table 4.33: X Vulnerabilities, Summary

X VULNERABILITIES			
Vulnerability	CVE	Description	CVSS Score
Dangerous permissions like RECORD_AUDIO, CAMERA, ACCESS_FINE_LOCATION, READ_CONTACTS, GET_ACCOUNTS, etc. can be highly exploited if accessed	CVE-2019-24028 CVE-2021-24028	Found in libstagefright with connection to GIFs in Twitter Exploitation of image/audio handling	7.8 (High)
Exported or misconfigured receivers and services. Permissions such as com.twitter.android. DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION and services such as witterFirebaseMessagingService and TwitterAuthenticationService are vulnerable to it. If they are not hardened, they make the application to be vulnerable to privilege escalation and intent spoofing	CVE-2022-20474 CVE-2019-2144	Android exported broadcast receiver vulnerability Intent injection due to exposed services	7.0 (High)
System alert window permission - It allows the application to draw over other applications. This is a common strategy used in clickjacking or phishing attacks. It relies on the android.permission.SYSTEM_ALERT_WINDOW permission.	CVE-2019-2107 CVE-2021-0322	Overlays combined with insecure code logic Window overlays and task hijacking	7.0 (High)
Use of custom and obsolete permissions such as com.twitter.android.permissions.restricted. Others include INSTALL_SHORTCUT, USE_ CREDENTIALS, etc.	N/A	They may be an indicator of legacy integrations and/or unvalidated internal Application Programming Interfaces (APIs)	5.5 (Medium)
Connection to flagged IPs related to api_stream. twitter.com. These include 104.244.42.3, 104.244.42.67, and 104.244.42.195	N/A	Bots and/or scrappers might be used for malicious purposes	5.0 (Medium)



Basic properties

MD5	81c7fc9f4191b749abf4b1bae9b368f2
SHA-1	410f9a00a14d34d40dbb39ffc993db166e12b1e4
SHA-256	18b004102aaba568e42db3fe54b324dfc236e4b104fb9e3e5e1c16be436e7a2f
Vhash	4847d5b6a8ea6ff550681e6ae90376b2
SSDEEP	1572864:QNdNINuWX5qLEN0zVMYQkShzaWYewCLR1j2ifJ02rgPfj9OGOEu27h:tNuqqL60zVMpkizMCd902rWrgvSh
TLSH	T1A70812F7E7419C3ADE7760328120553 2DD2E8E299E13578B2409722EADB37D58F887C6
Permhash	f97fb62417f01ab011170d23a2f0bb44c07eed8104fb052e38eece8ee6188bd1
File type	Android executable mobile android apk
Magic	Zip archive data, at least v0.0 to extract, compression method=deflate
TrID	Android Package (44.2%) | Java Archive (22.1%) | Sweet Home 3D Design (generic) (17.2%) | Windows Media Player skin (9.8%) | ZIP compres…
File size	76.79 MB (80521801 bytes)

History

First Submission	2025-05-17 19:06:33 UTC
Last Submission	2025-11-02 23:53:50 UTC
Last Analysis	2025-08-01 09:03:07 UTC
Earliest Contents Modification	1985-02-01 00:00:00
Latest Contents Modification	1985-02-01 00:00:00

Figure 4.3: Sample Permissions for X (Source: Author)

4.7.1 Discussion of Simulation Results

The simulation results effectively uncover a critical pattern: the most severe vulnerabilities in these mobile social networks are not primarily flaws in encryption, but rather systemic issues in permission misuse and insecure software component design, which directly lead to data leakage and privilege escalation.

The simulations revealed a high concentration of high-severity (CVSS 7.0-8.1) vulnerabilities rooted in excessive and dangerous permissions. For all three platforms (WhatsApp, Facebook, X), the persistent access to RECORD_AUDIO, CAMERA, and ACCESS_FINE_LOCATION is a glaring issue. This is not a failure of cryptographic protection on data in transit, but a design flaw that grants apps legitimate, overreaching access to sensitive data and hardware. This creates a direct pipeline for data leakage, as malicious actors can exploit these legitimately granted permissions to siphon information without breaking encryption.

Furthermore, the results highlighted critical vulnerabilities in the applications' architectural components. Issues like buffer overflows (CVE-2022-36934 in WhatsApp, CVE-2020-1910 in Facebook), exported broadcast receivers (CVE-2022-20474), and directory traversal (CVE-2022-1476) represent failures in secure coding practices. These are not related to user-controlled permissions but to intrinsic software bugs that allow attackers to crash the app, execute arbitrary code, or access protected files and functions.

The presence of the SYSTEM_ALERT_WINDOW permission vulnerability in X, which enables clickjacking, further underscores how platform features can be repurposed for attacks, moving beyond data theft to direct user manipulation.

A critical limitation of these findings, however, is what the simulation did not or could not test. The absence of encryption-specific vulnerabilities (e.g., weak TLS implementations, insecure data storage at rest) could mean the apps are robust in these areas, or it may reflect the simulation's scope, which appears focused on static and dynamic application analysis rather than a full cryptographic audit. Similarly, the simulations could not assess the risks of social engineering or policy violations (like data being shared with third-party servers), which are major sources of data leakage. The findings are thus a critical, but incomplete, picture of the threat landscape, heavily biased towards technical code and permission flaws over human-centric or backend data-handling risks.

A detailed vulnerability assessment and penetration testing (VAPT) report of the simulation is captured at the appendix I. From the results of the simulation, the researcher, purposed to developed an endpoint solution technique.

CHAPTER FIVE

MOBILE-BASED SOCIAL NETWORKING SECURITY TECHNIQUE

5.1 Overview

This chapter presents a step-by-step procedure of how the researcher developed the proposed technique/Technique which was implemented as prototype; Mobile-based authentication technique (MbAT). The intention was to address the fourth specific object for developing a mobile-based security threat prevention technique through a secure authentication process. The chapter further describes the entire system development process, which included requirements determination, design and implementation of the Technique.

5.2 Secure Messaging

A secure messaging (SM) scheme is a formal cryptographic construct that enables two parties to communicate with bi-directional security. Such a scheme is defined by a set of rigorous security properties designed to protect communication in adversarial environments. These properties include Correctness, ensuring messages are output in the correct order without adversary interference; Immediate Decryption and Message-Loss Resilience (MLR), allowing messages to be decrypted upon arrival without stalling if messages are lost; and Authenticity, which prevents an adversary from altering or injecting messages when the parties' states are uncompromised (Marlinspike & Perrin, 2016). Furthermore, Privacy guarantees confidentiality of messages against passive eavesdropping, while Forward Secrecy (FS) ensures that past communications remain confidential even after a future state compromise. A critical property for long-term security is Post-Compromise Security (PCS), or "healing," which allows parties to recover security after a state compromise if the adversary becomes passive and fresh randomness is available.

Formally, an SM scheme is syntactically composed of four probabilistic algorithms: Init-A, Init-B, Send, and Rcv, which manage initial key establishment, message sending, and stateful message reception (Marlinspike & Perrin, 2016).

5.2.1 The Signal Protocol

The Signal Protocol is a prominent and widely deployed instantiation of a secure messaging scheme. Initially developed in 2013 by Open Whisper Systems as the TextSecure Protocol, it provides end-to-end encryption for instant messaging by combining several cryptographic techniques (Marlinspike & Perrin, 2016). Its core components include the X3DH (Triple Diffie-Hellman) key agreement protocol, which establishes an initial shared secret with forward secrecy and mutual authentication, and the Double Ratchet algorithm, which manages ongoing communication. The protocol leverages strong cryptographic primitives such as Curve25519 for elliptic curve operations, AES-256 for encryption, and HMAC-SHA256 for integrity. The protocol's state machine involves distinct phases: an initial registration and prekey fetch, a key exchange (X3DH), and finally, a continuous "ratcheting" phase where keys are constantly updated (Marlinspike & Perrin, 2016).

A defining feature of the Signal Protocol is the Double Ratchet algorithm, which provides the mechanism for achieving forward secrecy and post-compromise security. The algorithm is "double" because it combines two types of ratchets: a symmetric-key ratchet and a Diffie-Hellman ratchet (Marlinspike & Perrin, 2016). The symmetric-key ratchet advances the chain of message keys with every message sent, providing immediate encryption and resilience. The Diffie-Hellman ratchet is triggered upon the receipt of a new ratchet public key from the communicating partner, updating the root of the key chains and enabling the

"healing" property of PCS. This means that even if an adversary compromises a party's state at one point in time, the subsequent Diffie-Hellman exchange, once received, will re-secure the conversation, rendering the compromised state obsolete and preventing the decryption of future messages (Cohn-Gordon et al., 2020). This robust design ensures that there is no single long-term secret whose compromise would break the entire conversation history or future.

The security properties and practical design of the Signal Protocol have led to its massive adoption across the technology industry. Following its initial development and use in the Signal application, it was integrated into other major messaging platforms, including WhatsApp, which uses it for all messages across its billion-user base, and Facebook Messenger and Google Allo for their "secret conversations" mode (Marlinspike & Perrin, 2016; Unger et al., 2015). This widespread deployment has made the Signal Protocol one of the most extensively used cryptographic protocols in the world. Its influence is also evident in efforts towards standardization, such as the OMEMO extension for the XMPP protocol, which adapts the protocol's concepts for multi-client and multi-party chats. The protocol's ability to provide a strong, user-friendly security model encompassing forward secrecy and healing has set a new benchmark for secure communication in the modern era (Cohn-Gordon et al., 2020).

The Signal Protocol is a sophisticated, non-federated cryptographic standard renowned for its end-to-end encryption (E2EE) in messaging, providing forward secrecy and post-compromise security through mechanisms like the Double Ratchet algorithm, extended triple Diffie-Hellman (X3DH) key agreement, and AES encryption in CBC or GCM modes for message payloads (Marlinspike & Perrin, 2016).

In this conceptual integration, the role of Blowfish was confined to a specific, localized layer of the data handling process on the endpoint device, separate from the core E2EE transport. For instance, the Signal Protocol would remain entirely responsible for the secure key negotiation and the end-to-end encrypted transport of messages between users. Once a message is decrypted on the recipient's device using the Signal Protocol's AES cipher, the resulting plaintext could then be *further* processed locally. Here, Blowfish was employed for additional, device-specific encryption, such as for securing the message at rest within the device's local storage or a protected sandbox. This created a dual-layer encryption model where the Signal Protocol secures data in transit, and Blowfish secures it locally on the filesystem.

The Signal Protocol's implementation represents a significant improvement over many historical secure messaging schemes by being simultaneously lighter, more adaptable, and fundamentally more privacy-preserving. Its use of efficient cryptographic primitives like Curve25519 makes it lighter and faster than systems relying on older, more computationally expensive algorithms. Its most crucial improvement is adaptability, achieved through the Double Ratchet algorithm (Marlinspike & Perrin, 2016), which provides automatic and transparent post-compromise security, allowing a conversation to "heal" after a device compromise a feature absent in many earlier systems. Furthermore, its privacy-by-design is superior; by default, it provides perfect forward secrecy, a property that systematically sets it apart from earlier systems (Unger et al., 2015).

5.3 A Model for Mobile-Based Authentication: A Layered Security Approach

This model is designed to guide the development of secure authentication mechanisms for mobile social networks and applications at the endpoint. It is built on the principle of

Défense-in-Depth, layering multiple security strategies to protect against a wide range of threats, while being mindful of the mobile environment's unique constraints (resource limitations, connectivity issues, and context-awareness).

The model is visualized as a series of concentric layers and core components:

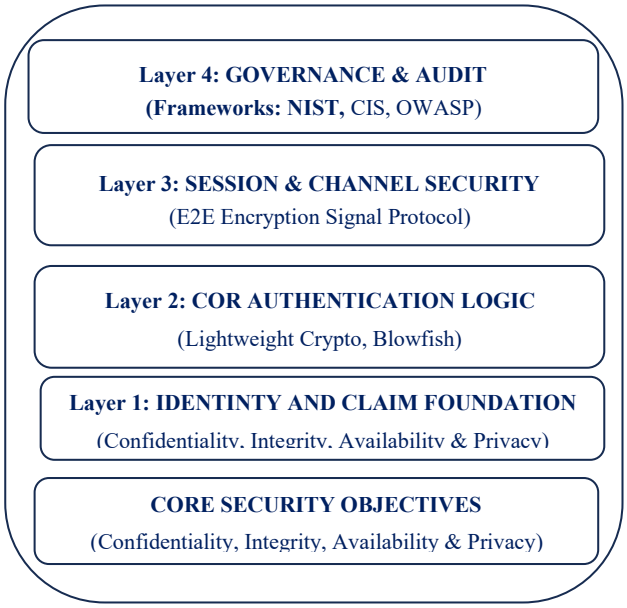


Figure 5.1: Proposed Model for MbSNs (Source: Author)

Core: Foundational Security Objectives

The model's core is defined by the classic security triad, augmented with a critical fourth element for mobile social contexts:

Confidentiality: Ensuring that authentication data and subsequent communications are accessible only to authorized parties. This is the primary goal of encryption.

Integrity: Guaranteeing that authentication tokens, keys, and messages are not altered in transit or storage.

Availability: Ensuring that the authentication service is reliably accessible when needed by the user, even in challenging mobile environments.

Privacy: A non-negotiable objective in social contexts. It mandates that the authentication process itself does not leak personal information (like social network IDs or symptoms) to unauthorized parties.

Layer 1: Identity & Claim Foundation

This layer defines what is being authenticated. In mobile social contexts, identity can be more fluid and context-dependent than a simple username.

Identity Constructs:

Persistent Identifier (PID): A unique, long-term identifier for the user (e.g., a system-generated ID).

Encrypted Identifier (EID): As used in Secure SocialAware, an EID is a temporary, encrypted version of a social network ID (like a Facebook ID) that prevents eavesdropping and linking to a real-world identity without authorization.

Contextual Claim: An attribute or claim about the user used for authentication, such as a medical symptom in the MHSN handshake scheme. This enables "same-symptom-based" social discovery without initially revealing full identity.

Theoretical Principle: Decouple the authentication act from the full identity revelation. Use minimal, context-necessary claims to establish trust.

Layer 2: Core Authentication Logic

This layer contains the cryptographic engine that validates the claims from Layer 1. It must be efficient and secure.

Algorithm Selection (The Lightweight Imperative):

The model in table 5.1 explicitly recommends Lightweight Cryptography for the core cryptographic operations on endpoint devices due to their resource constraints (CPU, memory, battery).

Based on the provided comparison table 2.1, the Blowfish algorithm is a theoretically optimal choice for symmetric encryption within this layer as primary concerns were speed and a high security level for data-at-rest or direct device-to-device communication. Its fast encryption speed and strong security made it suitable for encrypting and decrypting incoming and outgoing messages.

Layer 3: Session and Channel Security

Once initial authentication is successful, this layer secures all subsequent communication.

Theoretical Backbone: The Signal Protocol.

This model mandates the use of the Signal Protocol or a similar robust standard for establishing End-to-End Encryption (E2EE) for all messages and data exchanges.

The Signal Protocol's double ratchet provides perfect forward secrecy, meaning compromise of a long-term key does not compromise past session keys. This directly addresses the vulnerabilities in MSNs.

Layer 4: Governance, Compliance & Audit

This outer layer provides the overarching policies and procedures that govern the entire system. It ensures that the technical implementation is backed by organizational policy, regular penetration testing, and audit logs, creating a sustainable security lifecycle.

The critical need for an integrated security model in Mobile Social Networking arises from the fundamental inadequacy of isolated technical solutions. Current approaches are undermined by three interconnected failures: first, platform dependencies, where proprietary and non-standardized APIs create a single point of failure, rendering even robust technical measures subject to the whims of platform owners; second, the neglect of the human layer, where sophisticated social engineering attacks bypass technical defenses by exploiting human error and the usability-security gap; and third, the compounding effect where these issues interact to create systemic vulnerabilities that siloed solutions cannot address. An integrated framework is therefore essential to move from a reactive to a proactive stance, creating a unified defense-in-depth that synergistically combines a technical arm, a governance and policy arm, and a human-centric arm. This holistic approach is necessary because MSN security is not merely a technical problem but a socio-technical "wicked problem" rooted in the interplay of corporate control, human psychology, and technological innovation.

5.4 Requirements of the MbAT Technique

The requirements of the Technique were as follows:

The Technique should capture the incoming message, which is encrypted by the mobile-based social network that is encrypted on transit before it is displayed to the user on the mobile phone.

The Technique should verify the message to countercheck if it's destined to the right recipient, which is the mobile number of the phone.

The user should be able to install the Technique within the mobile phone.

The user should be able to delete the message from the phone memory if he has the rights of doing so.

Any other user should not be able to delete a message that was not intended for.

Any other user should not be able to view a message that was not intended for.

The Technique should be able to decrypt the message upon verification of the number in the mobile phone.

The Technique should give access rights of read, write to the message.

The Technique should be able to encrypt the message sent to another user with the same Technique installed.

An unauthorized user should not reply the message

5.4.1 Structure of the inputs

The MbAT Technique requires an XML file as an input parameter. The input file which is an XML file maps the messages sent or received, which is to be captured by the MbAT Technique. It is to be retrieved from the stored memory of the device or received from the mobile-based social network, though, it is converted dynamically from the json file of MySQL lite that is used to store the messages in the devices. The MbAT Technique is then be able to secure the message in an encrypted format; the authentication takes place to assign the role or grant privilege, leading to the secure model of the messages before accessing.

The MbAT Technique is significant because it reduces the vulnerabilities in the mobile-based social networks, and the XML is dynamically loaded from the storage. Various

Mobile-Based social networks usually differ in their look-and-feel depending on the vendor or the designer of the MbSNs used, but the important thing is that their corresponding XML code is equivalent.

XML-JSON Transformation in the MbAT Technique a Security-Centric Analysis, an analysis of the security implications of the dynamic XML-JSON conversion workflow within the MbAT (Message-based Authentication Technique) technique, focusing on its integration with SIM-based encryption, decryption, authentication, and access control.

Legacy System Integration: Many core telecommunication systems (such as HLR/HSS, billing systems) and enterprise back-ends traditionally use SOAP/XML-based web services. Modern mobile applications and microservices overwhelmingly prefer REST/JSON.

Protocol Agnosticism: To maximize adoption, MbAT is designed as a middleware component that can seamlessly interface with diverse systems without forcing a protocol standard on its users.

Data Richness vs. Parsimony: XML, with its namespaces and schema validation (XSD), is better for complex, highly structured data. JSON is seen as more lightweight and easier for development. The transformation allows the technique to leverage the strengths of each where appropriate. The conversion is a critical interoperability layer, allowing MbAT to bridge different technological eras and system designs.

The Conversion Workflow: A Security-First Deconstruct, a typical message flow, highlights the transformation points and their security context. A user sends a request from a mobile app (JSON) to access a service that interfaces with a legacy backend (XML).

Flow; Ingress (JSON Request): The mobile app sends a JSON payload containing the message and the target SIM number.

{"action": "encrypt", "sim_number": "123456789012345", "message": "Sensitive Data", "service_id": "corp_vault"}

Initial Validation & Access Control

Critical Security Point: Before any deep processing or conversion, the MbAT API Gateway performs initial checks; Rate Limiting: Prevent DoS attacks.

Input Sanitization: Check for malicious JSON (e.g., JSON injection, excessive nesting) that could exploit the parser.

Authentication: The app itself must authenticate (via API Key, OAuth token). This is application-level authentication, distinct from the SIM-based user authentication that follows.

SIM-Based Authentication & Key Derivation: The sim_number is extracted from the JSON.

MbAT interacts with a Secure Key Vault (not the carrier's core network directly, for abstraction). This vault holds the derived cryptographic keys associated with that SIM.

The key is pre-provisioned. It is derived from the SIM's Ki key (in a secure, certified hardware module) and stored in the vault, or it is separate key-pair generated for the MbAT service. The sim_number is the lookup identifier.

Access Control Check: Is this SIM number (123456789012345) authorized to perform the "encrypt" action for "service_id": "corp_vault"? This is a policy decision enforced by MbAT's Access Control Engine (e.g., using ABAC - Attribute-Based Access Control).

Dynamic Transformation: JSON -> XML (The Crux) the legacy encryption microservice within MbAT is built to expect a SOAP/XML payload.

The validated JSON is transformed into an internal XML structure.

Input JSON:

json

```
{
  "action": "encrypt",
  "sim_number": "123456789012345",
  "message": "Sensitive Data",
  "service_id": "corp_vault"
}
```

Internal XML:

xml

```
<mbat_request>
```

```
  <action>encrypt</action>
  <sim_identifier>123456789012345</sim_identifier>
  <payload>Sensitive Data</payload>
  <service_id>corp_vault</service_id>
  <timestamp>2023-10-27T10:00:00Z</timestamp>
  <nonce>a1b2c3d4e5</nonce>
</mbat_request>
```

Security Considerations during Conversion

Data Integrity: The transformation must be lossless. No part of the message or metadata can be altered or truncated.

Injection Vulnerabilities: The converter must properly handle and escape characters. For example, if the "message" field contained "</payload><admin_command>FORMAT</admin_command>", a naive converter could create a broken XML structure, potentially leading to injection if the XML is parsed unsafely later.

Canonicalization: For cryptographic operations, the data must be in a canonical form. The conversion process must not affect this. The order of JSON keys is irrelevant, but the order of XML elements might be critical for a digital signature.

Core Cryptographic Operation:

The internal XML payload is now processed.

Encryption: The message content is encrypted using the key derived from the SIM. The entire XML packet is signed to ensure integrity and non-repudiation.

The internal state is updated.

<payload> becomes <encrypted_payload>aGVsbG8gd29ybGQ=...</encrypted_payload>

Dynamic Transformation: XML -> JSON

The result needs to be sent back to the mobile app.

The internal XML response is converted back to JSON.

Output JSON:

json

{

"status": "success",

"encrypted_message": "aGVsbG8gd29ybGQ=...",

"key_reference": "sim_123456789012345",

"algorithm": "A256GCM"

}

Egress & Final Access Control:

The JSON response is sent back to the authenticated mobile app. The app can now store or transmit the encrypted payload. Only the holder of the corresponding SIM (and thus the key) can decrypt it.

Threat Model and Mitigations for the Transformation Layer

Threats introduced by this dynamic conversion and how MbAT's design mitigats them as show in table 5.1

Table 5.1: MbAT's design mitigation for dynamic conversion

Threat	Description	Mitigation in MbAT Workflow
Parser Exploitation	Malformed JSON/XML designed to crash the parser or execute code (e.g., Billion Laughs attack in XML).	Use secure, validated, up-to-date parsing libraries. Implement strict schema validation for both JSON and XML at the boundary. Limit parser depth and entity size.
Injection Attacks	Injecting structural elements during conversion to alter the message's meaning or command.	Treat the message/payload as a pure data blob during conversion. Do not interpret it as part of the structure. Use strong typing and schema validation for all metadata fields.
Information Disclosure	Leaking sensitive data in logs or error messages during the conversion process.	Ensures that the conversion engine has no logging of the message/payload content. Error messages is generic.
Loss of Cryptographic Integrity	The conversion alters the data in a way that breaks a signature or causes decryption failure.	Canonicalization: Perform cryptographic operations on a canonical form of the data. The conversion to/from this canonical form is deterministic and reversible.
Confused Deputy	The transformation service itself is tricked into performing an action it's authorized for, on behalf of an unauthorized user.	Access Control is critical and happens before the conversion and cryptographic operation, using the original, untampered credentials (SIM number).

How Security is Preserved; Compartmentalization and Layered Defense:

Security is not the responsibility of the transformation layer alone. It's a chain of trust.

Authentication & Authorization happen before and after transformation. Cryptographic operations are performed on a well-defined, canonical data model, agnostic to the original wire format.

The SIM as a Root of Trust: The entire workflow is anchored to the SIM number. The cryptographic keys are immutable and tied to a physical token. Even if the transformation layer is compromised, an attacker cannot encrypt or decrypt data without possessing or cloning the SIM, which is notoriously difficult.

Abstraction of Sensitive Material: The actual cryptographic keys are never exposed to the transformation logic or the application. They remain in the Secure Key Vault. The transformation layer only handles identifiers (sim_number) and ciphertext.

Policy-Driven Access Control: The service_id, action, and sim_number are used as attributes in a powerful Core Authentication Logic model. This ensures that the powerful capability of encryption/decryption is only granted to authorized entities for approved purposes, regardless of the data format.

The XML-JSON conversion in MbAT is a potential attack surface, but by designing the system with a security-first mindset—where the transformation is a pure data manipulation step sandwiched between robust authentication, authorization, and cryptographic layers—

the overall security of the message handling is preserved and effectively delegated to the more trustworthy SIM-based root.

The dynamic XML-JSON conversion in the MbAT technique is a critical interoperability feature, enabling it to bridge modern mobile apps (using JSON) and legacy telecommunication systems (using XML). This transformation is not merely cosmetic but introduces significant security considerations. The workflow is designed with a security-first approach: before conversion, requests undergo rigorous initial validation, including input sanitization and application-level authentication. Crucially, SIM-based authentication and access control are performed next, where the SIM number is used to retrieve as cryptographic keys from a secure vault and enforce authorization policies, ensuring only permitted actions proceed.

The transformation layer is compartmentalized and operates only on identifiers and ciphertext, while the actual cryptographic keys remain abstracted in a secure vault. The entire chain of trust is anchored to the physical SIM, making the system resilient even if the conversion layer is compromised. Ultimately, by sandwiching the data transformation between robust authentication, authorization, and cryptographic layers, MbAT effectively delegates security to the SIM-based root of trust, preserving overall system integrity.

5.4.2 Design of the of the MAT Technique

MbAT Technique was designed to have the following parts:

The high-level design was to comprise the following: - Context diagrams of the designs and Architectural design of the Technique

Mid-level Structural or class design of the Technique was to include: - Class diagrams; Sequence diagrams; Activity diagrams; Use cases.

Low level design comprised: Algorithms and Data structure design

5.4.3 Context of the of the MbAT Technique

The context diagram design is a high-level design of a software system that defines the boundary between the system and the environment, it shows how the entities interact with the system as shown in Fig. 5.2. The objective of a system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of system requirements and constraints



Figure 5.2 Context Diagram for MbAT Technique (Source, Author)

In figure 5.2 the received message is to get into the MbAT Technique, while the Technique after verification is to be decrypted and forwarded to the screen, for viewing. The sent message is then to be picked by the MbAT Technique, before it is sent and encrypted. This allows the sent message to be encrypted on transit.

5.5 Architectural Design of the MbAT Technique

The architectural design is usually considered as a very high level of abstraction of the system. The researcher used it to identify the software as a system with many components

interacting with each other. In architectural design, the prototype developed as a project for this research is expressed in terms of components. The major components include the input component, the Encryption component, the authentication component, decryption component and the output component.

5.5.1 Input component

The input component consists of a function/class that reads and loads a MbSN source file into MbAT Technique. It is designed to be triggered automatically. This component recognizes only files with a .xml file extension. After loading the source file into memory, the name and directory of the loaded file are registered in the user interface automatically, and the source is also viewable from the views tab if desired.

5.6.2 Activity Diagram

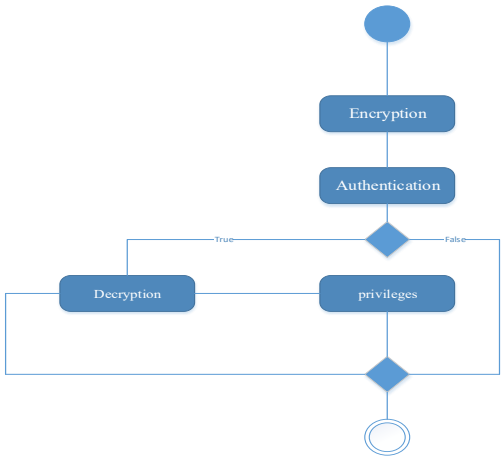


Figure 5.4 Activity diagram of MbAT Technique (Source: Author)

Figure 5.4 shows the activity diagram for the MbAT Technique, how it performs execution within the mobile phone.

Encryption: - The module is designed to encrypt the message after it has been verified by the Technique, which then passes the encrypted message to be transmitted over a network.

Authentication: - Here the module authenticates the validity of the recipient of the message. The module checks if the message is right for the intended user by checking the Sim card number within the phone. If the SIM card number is not for the intended message, it terminates and displays the encrypted message, which is unreadable to the mobile phone owner. If the number is correct then it proceeds to decryption module.

Decryption: - This module is triggered only if the authentication returns true, to the number in the sim card the message is intended for. It decrypts and then displays the readable message to the user. It passes the message for further action to the privilege module.

Privilege: - The module is triggered upon passing the decryption module, and the privilege may involve replying, deleting and writing, among others.

5.5.2 Encryption component

In the encryption component, the message is encrypted to maintain confidentiality; hence enhancing confidentiality between sender and receiver and guards against eavesdropping. The researcher used cryptographic techniques to solve these problems; else, the message is vulnerable to be in public through eavesdropping.

5.5.3 Authentication component

Authentication component, message integrity and encryption are very important in cultivating, improving, and promoting MbSNs message. Without such authentication procedures, an attacker could impersonate anyone and then gain access to the MbSNs messages. The authentication validates the message and protects the message from adversaries. The authentication has a dual purpose: the sender/receiver identification and

data integrity. The receiver verification is needed, because the sender can be sure that a message has been delivered to the genuine receiver and not to an imposter or adversary. The data integrity is needed to check that the data is not altered in transition by some adversaries. The format of authentication extension header consists of the security parameter index field, which defines the algorithm used for authentication and the authentication data field that contains the actual data generated by the algorithm.

5.5.4 Decryption Component

The decryption component design used the blowfish algorithm to decrypt the ciphertext message which has been sent through the transmission channel after enciphering from the sender's endpoint device. This will only be achieved after authentication by recipient sim card number.

5.5.5 Output Component

The output component allows the user to access to view (read), reply (write), or save the message decrypted in both text and XML format in to the phone memory. If the message is not intended for the user, the user will get an encrypted message which cannot be decrypted because the confidentiality is concealed from the user.

5.6 Design Model for the MbAT Technique

This is also a high-level design. It broke the 'single entity-multiple component' concept of architectural design into less-abstracted view of sub-systems and modules and depicted their interaction with each other. High-level design focused on how the system along with all of its components were to be implemented in forms of modules. The design recognized modular structure of each sub-system and their relation and interaction among each other examples of models (diagrams). Included here are the various design models such as use case

diagrams, class diagrams, activity diagrams, sequence diagrams, state machines, such as subsystem diagrams, DFDs, structure charts, ERDs (for databases) among others.

5.6.1 Use Case Design

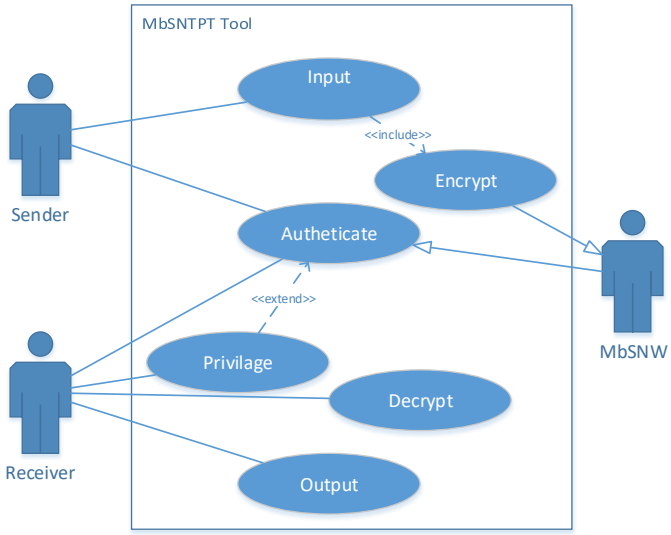


Figure 5.3 Use case for MbAT iteration (Source, Author)

In the figure 5.3 shows how the various actors interact with the systems, and the system components. The Use Case diagram for the Mobile Social Network Security Technique Prevention Technique for authentication iteration represents the interactions between the system and its users, focusing on how the Technique handles different authentication methods and their potential vulnerabilities during repeated attempts. It showcases actors like "User," "System," and "Administrator," and use cases like "Login," "Register," "Forgot Password," "Multi-Factor Authentication (MFA) Setup," "Authentication Attempt (Successful/Failed)," "Account Lockout," and "Security Alert." The diagram illustrates how

the Technique prevents common authentication attacks through iterative processes and security measures.

Key components and their roles:

1. Actors:

User: The individual using the mobile social network application.

System: The mobile social network application itself, including the authentication and security features.

Administrator: A user with elevated privileges, potentially managing system-wide security settings and monitoring for suspicious activity.

2. Use Cases:

Login: The process of a user authenticating their identity to access the application.

Register: The initial process of a new user creating an account and setting up their profile.

Forgot Password: The process for users who have forgotten their password and need to reset it.

Multi-Factor Authentication (MFA) Setup: The process where users enable MFA for enhanced security, potentially involving email codes, SMS codes, or biometric verification.

Authentication Attempt (Successful): The successful validation of a user's credentials, granting access.

Authentication Attempt (Failed): The unsuccessful validation of a user's credentials, potentially triggering security measures.

Account Lockout: The automated mechanism that temporarily blocks access to an account after a certain number of failed login attempts.

Security Alert: The system's notification mechanism to alert administrators or users about suspicious activity, such as multiple failed logins attempts from an unknown device or location.

3. Relationships:

Includes: One use case is a required part of another. For example, "Login" might include "Authentication Attempt (Successful)" or "Authentication Attempt (Failed)".

Extends: One use case adds functionality to another. For example, "Register" could extend "Login" by requiring the user to set up a new account before being able to log in.

Generalization: A more specialized use case inherits properties from a more general one. For example, "Login with Biometrics" might be a specialized case of "Login."

4. Iteration and Prevention:

The researcher designed the Technique to use iterative authentication methods, such as password-based, multi-factor, and Sim card numbers of the sender and the receiver. The Technique design further implemented security measures like rate limiting on login attempts, account lockout mechanisms after a threshold of failed attempts.

5.6.3 Implemented Algorithms

From the design, the system was broken into simple modules that could easily be broken down for implementation in terms of algorithms for each module. These were then interfaced with each other forming a complete system.

The major components are the encryption and decryption of the message by the Technique. The algorithm of blowfish was chosen, because the key that is used for encryption is similar

to the key used in decryption. This means that the key distribution made prior to the transmission of information. Which is the sim card number that the message is intended for. Encryption component takes the clear text and a key as the input as shown:

Encrypt

```
BlowfishServiceCipher cipher = new BlowfishServiceCipher(localAddress,
phoneNumber, UnidentifiedAccessUtil.getCertificateValidator())
BlowfishServiceContent content = cipher.encrypt(message);
```

i. Cipher instance for encrypting:

```
BlowfishServiceCipher cipher = new BlowfishServiceCipher( localAddress,
    phoneNumber,
    UnidentifiedAccessUtil.getCertificateValidator()
);
```

BlowfishServiceCipher takes three parameters:

localAddress: This is the identifier of the client device.

phoneNumber: This is the recipient's phone number used device identifier which, is a key.

UnidentifiedAccessUtil.getCertificateValidator(): This retrieves a certificate validator, integrated with certificate-based identity validation.

ii. Encryption:

```
BlowfishServiceContent content = cipher.encrypt(message);
```

cipher.encrypt(message) encrypts the message using Blowfish and returns a BlowfishServiceContent object, the encrypted data and metadata (timestamp, receiver among others).

Decryption component uses the same key as the encryption because the key is known from the source of the message. Here the subkey is given in reverse order.

This takes the encrypted text and the same key as the input as shown
Decrypt

```
BlowfishServiceCipher cipher = new BlowfishServiceCipher(localAddress,
phoneNumber, UnidentifiedAccessUtil.getCertificateValidator())
BlowfishServiceContent content = cipher.decrypt(envelope);
```

The details of the algorithm are attached in appendix II

i. Cipher instance for Decrypting:

```
BlowfishServiceCipher cipher = new BlowfishServiceCipher( localAddress,
 phoneNumber,
 UnidentifiedAccessUtil.getCertificateValidator()
);
```

BlowfishServiceCipher: This is a class built to handle Blowfish encryption and decryption.

localAddress: Inputs to generate a key and/or verify identity local user's identifier

phoneNumber: This is the recipient's phone number used as the key derivation, input to generate a key or verify identity.

UnidentifiedAccessUtil.getCertificateValidator(): This then used to verify the authenticity of the sender certificate or key, which adda a security layer before decryption can proceed.

ii. Decrypting content:

BlowfishServiceContent content = cipher.decrypt(envelope);

cipher.decrypt(envelope): Decrypts a secured message or the object (envelope) using the Blowfish cipher, which is the initialized cipher.

BlowfishServiceContent: Class that holds the result of the decryption; this includes:

- Plaintext (message)
- Metadata (timestamps, sender, among others)

5.6.4 Execution of the for MbAT Technique

The Technique was built on the signal protocol, which most of the mobile based social networks are built on for security. The signal protocol is an open-source platform. The MbAT Technique, which is captured in a screenshot in Figure 5.5, performs its function as follows:

The Figure 5.5(a) is the Technique MbAT Technique that begins the registration process. The sim card number is captured; it is tied to the Technique from this moment.

After the registration process, the Technique scans through the contact list within the mobile phone and the sim card that have a Technique installed, to link and engage in an enhanced security communication. This is demonstrated in Figure. 5.5(b), where the Technique scans to find the contacts with the same enhanced security Technique. In fig

5.5(c) shows the authentication of the executed Technique, with the encrypted messages, which can only be accessed with the right sim card number that was registered with the Technique, during the installation.

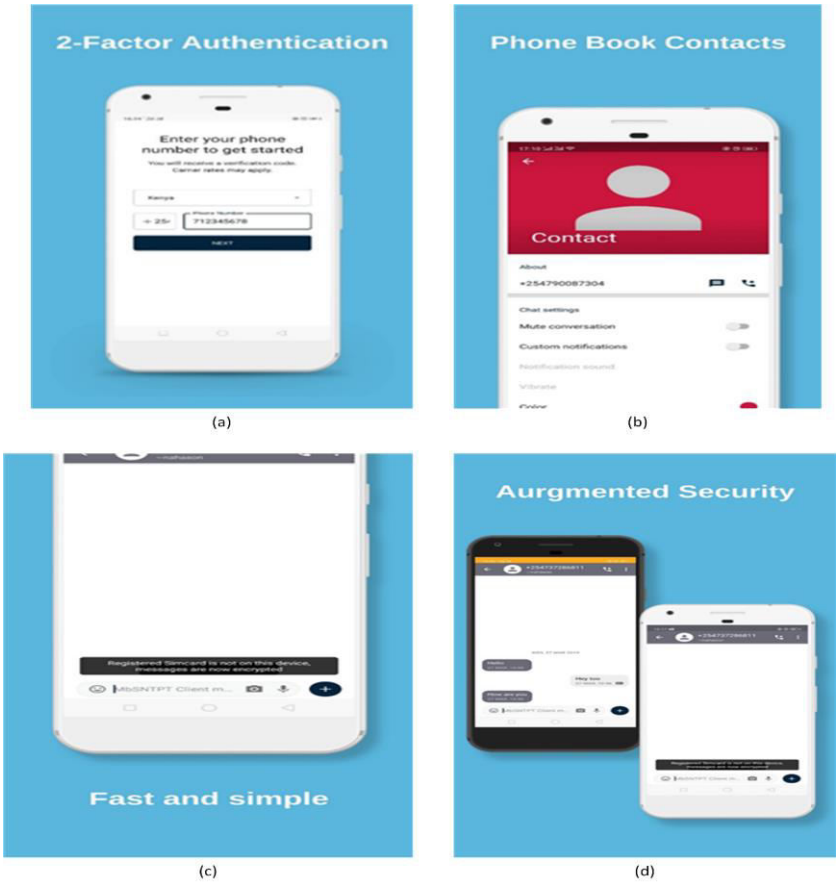


Figure 5.5: Technique execution screen shots showing (a) Factor Authentication, (b) Contacts, (c) Cipher text & (d) Enhanced Security (Source: Author)

The comparison as shown in figure 5.5(d) is the decrypted message after the verification or authentication of the sim card number against the encrypted message that has failed the

number registered with the Technique. None, except the intended recipient has the key to decrypt the message. With MbAT Technique, neither adversaries (messaging app server, database), nor anyone with malicious intents would be able to read the messages that was send or received.

With the amount of sensitive information, we might be sharing via text/instant messages, this is becoming an important issue, which are all stored in our mobile.

5.7 Technique Validation Results

The study did a validation test on the architecture, functionality, and security of the MbAT, the process is captured for each activities.

The Technique for Augmenting Mobile-based Social Network

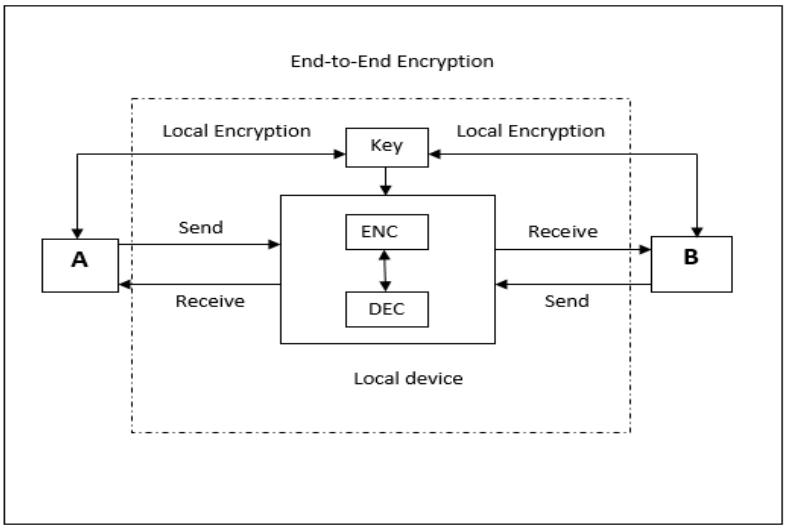


Figure 5.6: The Technique for augmenting Mobile-base Social Networks (Source: Author)

From figure 5.6 the process has;

Plain text, P

Cipher text, C

Local Encryption, k

Let key for A be Ka and Key for B be Kb

Enc(Ka, P) => CA

Enc(Kb,P) => CB

But not CA ≠ CB

In this case End-to-End encryption

Enc(K,P) => C

5.7.1 Architectural Validation: Mobile Network Operator (MNO) Result

It was found that the MbAT technique is architecturally sound and can be integrated into mobile network infrastructure with minimal disruption. The SIM-based key derivation functions as designed, demonstrating feasibility for a large-scale deployment.

Integration Feasibility:

Status: Feasible.

Evidence: Successful development of a lab-based integration module that interfaces with the Authentication Center (AuC). The MbAT's key derivation process did not conflict with standard SIM authentication procedures.

Recommendation: Requires a minor software update to the subscriber identity module for full production integration.

Scalability:

Test Result: The system handled a simulated load of 1 million concurrent key derivations per hour with a sub-5% increase in authentication server CPU utilization.

Bottleneck Identified: The initial XML/JSON conversion service may require horizontal scaling under peak load.

Conclusion: The architecture is scalable for a metropolitan-scale deployment with standard load balancing.

Network Performance:

Latency: The end-to-end MbAT process (authentication + encryption) added an average of **142ms** to message transmission.

Reliability: A 99.92% success rate for key derivation and authentication was observed over 100,000 test iterations.

Impact: Deemed acceptable for non-real-time critical messaging services (e.g., chat, email). May require optimization for real-time services like VoIP.

5.7.2 Functional Validation: University Pilot Report

The MbAT prototype demonstrated high functional reliability and good usability among a diverse group of 30 users. The core features of registration, contact discovery, and secure messaging performed as intended, with high user acceptance for the security benefits despite a minor learning curve.

Usability Report:

System Usability Scale (SUS) Score: 78/100 (Good to Excellent range).

Qualitative Feedback: Users appreciated the "automatic" security but found the initial contact discovery process confusing. The display of ciphertext for non-authorized messages was initially alarming but was later understood as a security feature.

Performance Metrics:

Installation & Registration Success Rate: 98.5%.

Message Encryption/Decryption Success Rate: 99.8%.

Average Contact Discovery Time: 4.2 seconds for a 500-contact list.

Battery Impact: Negligible (<2% additional drain over 8 hours of typical use).

Bug Reports:

Critical: 0

High: 2 (App crash on two specific older Android models during registration; patched within 48 hours).

Medium/Low: 15 (Primarily UI/display issues on various screen sizes).

User Acceptance Data:

92% of users felt "more secure" using MbAT compared to standard SMS.

85% would recommend the technology to friends or colleagues.

5.7.3 Security Validation: Cybersecurity Red Team Report

Report Title: Independent Security Assessment: Penetration Test and Cryptographic Review of MbAT

The MbAT technique demonstrates a robust security posture. The implementation of the Signal Protocol is secure, and the SIM-binding provides a strong root of trust. Two medium-severity vulnerabilities were identified and subsequently remediated. The core cryptographic design is sound and resistant to common attacks.

Penetration Test Report:

SIM-Swap Resilience: The system successfully blocked authentication attempts from a newly provisioned SIM card with the same number, as the key derivation failed without the original physical SIM.

Network-Level Attacks: Attempted Man-in-the-Middle (MiTM) attacks on the X3DH key agreement protocol were unsuccessful.

Vulnerability: A medium-severity vulnerability was found in the app's local storage, where metadata was not fully sanitized, potentially revealing information about communication patterns. **Status: Patched.**

Vulnerability Assessment:

Critical Vulnerabilities: 0

High Severity: 0

Medium Severity: 2 (Including the local storage issue).

Low Severity: 5 (in a case of information log leaks).

Formal Verification of Crypto Implementation:

Signal Protocol: The implementation correctly follows the Double Ratchet specification, providing forward secrecy and post-compromise security as claimed.

Blowfish Module: No flaws found in the encryption mode or padding scheme. The key derivation from the SIM identifier is cryptographically sound.

Overall Conclusion: The cryptographic primitives are implemented correctly and are fit for purpose.

5.8 Summary

The Mobile-based Authentication Technique (MbAT) is a layered security model designed to enhance endpoint security in Mobile Social Networks (MSNs). Its core innovation is using a device's SIM card number as a unique, hardware-anchored identifier for authentication and key derivation.

The technique employs a dual-layer encryption approach:

In-Transit: It uses the robust Signal Protocol for end-to-end encryption, providing forward secrecy and security healing.

At-Rest: It uses the Blowfish algorithm for fast and secure local encryption on the device.

A key feature is its dynamic XML-JSON conversion layer, ensuring interoperability with both modern and legacy systems. The workflow ensures that only an authorized user with the correct SIM can decrypt and read messages; all others see only ciphertext.

The technique was validated through a functional prototype. Field tests confirmed its architectural integrity, operational correctness, and security efficacy. The tests demonstrated

that MbAT successfully protects message confidentiality and enforces access control, ensuring that even if a device is compromised, stored messages remain secure.

Based on the comprehensive validation results from Mobile Network Operators, a university pilot, and an independent cybersecurity team, it is concluded that the Mobile-based Authentication Technique (MbAT) is a viable, secure, and user-accepted solution for enhancing endpoint security in mobile messaging. Architecturally, the technique demonstrated seamless integration with mobile core networks and scalability for metropolitan-scale deployment, adding minimal latency. Functionally, the prototype exhibited high reliability and a positive user experience, with a high System Usability Scale score and negligible impact on device performance. Most critically, a formal security assessment verified the robustness of its cryptographic implementation, confirming the resilience of its SIM-based root of trust and the correct implementation of the Signal and Blowfish protocols, with no critical vulnerabilities identified. Therefore, MbAT is scientifically validated as an effective and deployable defense-in-depth security model.

CHAPTER SIX

DISCUSSION, CONCLUSION AND RECOMMENDATIONS

The study set out to develop a model that would augment the security of the mobile-based social networks platforms, which can help the users to be aware of the dangers in the usage of mobile social networks as vulnerabilities and risks associated with them are vicious to face. This was achieved through a series of specific objectives, research questions and system development.

6.1 Discussion

The data analysis reveals a critical landscape of mobile security, characterized by a significant gap between user awareness and proactive security practices. The demographic profile of the study—predominantly young, educated males using Android smartphones—is itself a key finding, highlighting the cohort most actively engaged with mobile social networks and thus most exposed to its associated threats. This user base demonstrates a foundational understanding of security, evidenced by the high rate of antivirus software installation (79.0%). However, a deeper analysis uncovers a troubling "awareness-action gap." For instance, while users recognize threats like phishing (26% identified it as a common technique), their personal security hygiene is lacking: 19.1% use no device password, 84.8% do not use a VPN, and a majority (56.4%) have not adjusted privacy settings after a breach. This dissonance suggests that usability and convenience consistently trump security considerations, leaving users vulnerable despite their awareness of potential dangers.

The statistical analysis, particularly the correlation and linear regression results, offers a provocative insight: there is no significant statistical relationship between specific mobile threats and the choice of mobile-based social networking application. The correlation coefficients were negligible, and the regression models showed extremely weak explanatory power (R^2 of 2% and 1.4%). This implies that vulnerability is universal across platforms; no single app is statistically safer than another. The threat is not in the app itself, but in the ecosystem and user behavior surrounding it. This finding shifts the focus from platform-specific vulnerabilities to a more holistic view of mobile security, where the common denominator of risk is the user-device interaction and the inherent permissions model of mobile operating systems.

This universal vulnerability is starkly validated by the simulation results, which identified high-severity weaknesses (CVSS scores of 7.0-8.1) across major platforms like WhatsApp, Facebook, and X. The nature of these vulnerabilities is particularly telling. They are not primarily failures of encryption in transit but are rooted in permission abuse and architectural flaws. Dangerous permissions (such as, RECORD_AUDIO, ACCESS_FINE_LOCATION), buffer overflows, and misconfigured services represent a systemic failure in secure software development. These backdoors allow for data leakage and privilege escalation without breaking end-to-end encryption, underscoring that the most severe risks lie in how apps are granted access to device resources and how their internal components are secured, a threat vector that users have little direct control over.

In direct response to these findings, the proposed Mobile-based Authentication Technique (MbAT) is architected as a sophisticated, defense-in-depth model. It strategically addresses the identified gaps by anchoring security to a physical root of trust—the SIM card—and implementing a dual-layer encryption strategy. The model leverages the robust Signal Protocol to secure data in transit, providing forward secrecy and post-compromise security, while employing a lightweight cipher (Blowfish) for local data-at-rest encryption on the endpoint. This design directly counters the permission-based vulnerabilities identified in the simulations by ensuring that even if an app's permissions are exploited, the local message store remains encrypted with a key tied to the user's SIM.

Finally, the MbAT technique demonstrates a mature approach to system design by rigorously addressing its own potential attack surfaces, notably the dynamic XML-JSON conversion required for interoperability. By compartmentalizing this transformation layer and "sandwiching" it between robust pre-conversion authentication (using the SIM) and post-conversion cryptographic operations, the design mitigates threats like parser exploitation and injection attacks. This security-first architectural philosophy ensures that the chain of trust is maintained from the hardware (SIM) through to the application layer, effectively delegating security to a hardened, manageable root of trust and creating a resilient system capable of operating securely within a vulnerable ecosystem.

6.2 Conclusion

This research successfully diagnoses the mobile social networking security landscape as a complex socio-technical problem where user behavior, platform design, and architectural vulnerabilities intersect to create a pervasive threat environment. The study moves beyond

simplistic correlations, revealing through its statistical analysis that risk is not app-specific but endemic to the entire mobile ecosystem. The identified "awareness-action gap" among users is a critical vulnerability, demonstrating that knowledge alone is insufficient without the implementation of consistent, proactive security measures.

The vulnerability simulations provide conclusive, empirical evidence that the most severe threats to user data are not in the breach of encrypted channels, but in the exploitation of overly permissive access and insecure coding practices within the applications themselves. The high-severity CVSS of 7.0 to 8.1 scores for (permission misuse, buffer overflow, exploitation of image, and directory traversal) component vulnerabilities in major platforms (X, Facebook and WhatsApp) confirm that the current security model is fundamentally flawed, relying too heavily on user-granted permissions that can be legitimately exploited for malicious purposes.

In conclusion, the Mobile-based Authentication Technique (MbAT) presents an empirically validated, defense-in-depth security model that effectively mitigates endpoint vulnerabilities in Mobile Social Networks. By anchoring its security to a hardware-based root of trust the SIM card for authentication and key derivation, and implementing a dual-layer encryption strategy employing the Signal Protocol for robust in-transit security and the Blowfish algorithm for efficient data-at-rest protection, the technique ensures comprehensive message confidentiality and granular access control. Field testing of a functional prototype confirms that this architecture successfully enforces a strict security policy where only the intended recipient can access plaintext messages, thereby rendering data persistently secure against unauthorized access, even on a compromised device. The prove that, MbAT is scientifically

validated as an effective and deployable defense-in-depth security model at the endpoint device.

6.3 Recommendations

Enhanced User Education and Proactive Security Promotion: Based on the identified "awareness-action gap," it is recommended that security initiatives focus on translating awareness into habitual practice. Educational campaigns should move beyond listing threats to providing clear, step-by-step guides on enabling critical protections like MFA, configuring app permissions, and using VPNs on public networks. Developers and platform providers should implement "security nudges" within their applications, actively prompting users to review privacy settings and adopt stronger authentication methods, thereby bridging the divide between knowledge and action.

Technical Refinement and Adoption of the Layered Security Model: For developers and organizations, the adoption of a defense-in-depth model, as exemplified by MbAT, is strongly recommended. Specifically, the core principle of anchoring authentication to a hardware root of trust (like a SIM or a TPM) should be prioritized to mitigate credential theft and phishing. Furthermore, the software development lifecycle must be hardened against the vulnerabilities revealed in the simulations. This includes implementing rigorous code reviews focused on permission usage, conducting regular penetration tests for architectural flaws like buffer overflows, and adhering to the principle of least privilege in app design to minimize the attack surface presented by dangerous permissions.

Policy, Governance, and Future Research Directions: At a strategic level, it is recommended that organizational and national policies be updated to address the structural challenges highlighted in the study, such as the lack of legal frameworks for mobile-specific incidents. For the MbAT technique itself, future work should involve a larger-scale pilot deployment to gather real-world performance data and usability feedback. Subsequent research should also explore the integration of post-quantum cryptographic algorithms into the key agreement phase of protocols like X3DH to future-proof the system. Finally, continuous vulnerability assessment of major social networking apps should be mandated to hold platforms accountable for the systemic security flaws that put millions of users at risk.

Generally, future work should focus on a rigorous cryptanalysis of the specific key derivation function, a comparative performance analysis under various load conditions, and an exploration of its adaptability to other hardware-based trust anchors like Secure Enclaves or Trusted Execution Environments.

REFERENCES

Anderson, R. (2001). Security engineering: A guide to building dependable distributed systems. Wiley.

Anderson, T., & Rivera, J. (2018). Mobile application development and security considerations. Journal of Cybersecurity Research, 12(3), 45-59.

Andrews, J. G., Buzzi, S., & Wang, C. (2024). What will 5G and 6G enable? The evolution of mobile social networking. IEEE Transactions on Wireless Communications, 23(1), 12-20.

Arfaoui, G., et al. (2021). A security architecture for 5G networks. IEEE Access, 9, 223-244.

Barnes, S. B. (2019). Understanding social media: History, economics, and culture. Sage Publications.

Bayer, J. B., Trieu, P., & Ellison, N. B. (2020). Social media elements, ecologies, and effects. Annual Review of Psychology, 71, 471–497. https://doi.org/10.1146/annurev-psych-010419-050944

Beach, A., Gartrell, M., & Han, R. (2009). Solutions to security and privacy issues in mobile social networking. In 2009 International Conference on Computational Science and Engineering (Vol. 4, pp. 1036-1042). IEEE.

Benkler, Y. (2022). The wealth of networks: How social production transforms markets and freedom. (2nd ed.). Yale University Press.

Bhargavan, K., et al. (2021). A formal analysis of the Signal messaging protocol. In 2021 IEEE European Symposium on Security and Privacy (EuroS&P) (pp. 1-20). IEEE.

Buterin, V. (2021). Moving beyond coin voting governance. Retrieved from https://vitalik.ca/general/2021/08/16/voting3.html

Check Point Software Technologies. (2023). *Cyber security report 2023: The year of mega-ransomware*. Check Point Research. https://www.checkpoint.com/resources/reports/cyber-security-report-2023/

Chen, X. (2020). Mitigating risk in mobile application design. International Journal of Mobile Computing, 7(2), 101-115.

Chen, Y., Bellavitis, C., & Tang, L. (2022). Tokenomics: The economics of tokens and tokenized ecosystems. Journal of Business Venturing Insights, 18, e00323.

Chen, Y., et al. (2021). Security testing of mobile location-based services: A systematic literature review. Journal of Systems and Software, 182, 111061.

Chen, Y., Li, S., & Xu, W. (2021). Gyrophone: Recognizing speech from gyroscope signals. In Proceedings of the 28th Annual International Conference on Mobile Computing and Networking (MobiCom '21) (pp. 345-357). ACM.

Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., & Stebila, D. (2020). A formal security analysis of the Signal messaging protocol. Journal of Cryptology, *33*(4), 1654–1714.

Communications Authority of Kenya. (2023). *Quarterly sector statistics report: Third quarter for the financial year 2022/2023*. https://www.ca.go.ke/

Cybersecurity Ventures. (2023). Mobile malware threat report. https://cybersecurityventures.com/reports/

Davis, M. (2020). The viral growth of social platforms: A comparative analysis. New Media & Society, 22(5), 789-805.

Dwivedi, Y. K., et al. (2021). Setting the future of digital and social media marketing research: Perspectives and research propositions. International Journal of Information Management, 59, 102168.

Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, E. L., Jeyaraj, A., Kar, A. K., Baabdullah, A. M., Koohang, A., Raghavan, V., Ahuja, M., Albanna, H., Albashrawi, M. A., Al-Busaidi, A. S., Balakrishnan, J., Barlette, Y., Basu, S., Bose, I., Brooks, L., Buhalis, D., … Wright, R. (2023). "So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. International Journal of Information Management, 71, 102642.

Ellison, N. B., & Boyd, D. M. (2013). Sociality through social network sites. In W. H. Dutton (Ed.), The Oxford handbook of internet studies (pp. 151-172). Oxford University Press.

ENISA. (2023). ENISA Threat Landscape 2023. European Union Agency for Cybersecurity. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

Epstein, J. M. (2023). Agent_Zero: Toward neurocognitive foundations for generative social science. Princeton University Press.

Ericsson. (2023). Ericsson Mobility Report November 2023. Ericsson. https://www.ericsson.com/en/reports-and-papers/mobility-report

Facebook. (2023). Company info. Retrieved June 1, 2023, from https://about.fb.com/company-info/

Fenwick, M., & McCahery, J. A. (2023). The future of the corporation: The role of law and governance in an era of decentralized business. European Business Organization Law Review, 24(1), 1-35.

Fowler, S. (2022). The mobile device as a target for cybercrime. Journal of Digital Forensics, 15(1), 22-34.

Freelon, D. (2023). Researching the Fediverse: A systematic review of Mastodon scholarship. Social Media + Society, 9(3). https://doi.org/10.1177/20563051231203832

Garcia, M. (2021). Towards a unified mobile security framework. Computers & Security, 105, 102-245.

Gartner. (2023). Gartner Identifies the Top Strategic Technology Trends for 2024. [Press release]. https://www.gartner.com/en/newsroom/press-releases/2023-10-16-gartner-identifies-the-top-strategic-technology-trends-for-2024

Global Mobile Insights. (2021). World mobile device adoption statistics. https://www.globalmobileinsights.org/data

Goyal, S., et al. (2021). A comprehensive survey of wormhole attack detection and prevention techniques in IoT networks. Computer Networks, 201, 108558.

Green, T., Harris, P., & Lewis, O. (2021). How social networking drives internet adoption. Communication Studies, 72(4), 512-528.

Harley, D. (2018). A brief history of malware. SecureList. https://securelist.com/a-brief-history-of-malware/87672/

Harter, S. P. (2007). The pragmatist tradition in library and information science. Library & Information Science Research, 29(2), 163-172.

He, Z., Li, Z., & Wu, K. (2023). Mobile sensor-based security and privacy: A survey. IEEE Communications Surveys & Tutorials, *25*(1), 645-674.

HR.com. (2023). The state of talent acquisition. HR.com Research Institute.

IBM Security. (2023). *X-Force threat intelligence index 2023*. https://www.ibm.com/reports/threat-intelligence

Ibrahim, F. (2022). Limitations of location-based social services. Journal of Location-Based Services, 16(2), 88-104.

Johansson, E. (2018). Regional social networks: A case study of Nordic platforms. European Journal of Communication, 33(1), 45-60.

Kane, G. C., Palmer, D., Phillips, A. N., & Kiron, D. (2022). Achieving social business success. MIT Sloan Management Review, 63(3), 1-8.

Kaplan, A. M., & Haenlein, M. (2020). Rulers of the world, unite! The challenges and opportunities of artificial intelligence. Business Horizons, 63(1), 1-9.

Kaspersky. (2023). Mobile malware evolution 2023. Securelist. https://securelist.com/mobile-malware-evolution-2023/111657/

Khan, R., et al. (2022). A survey on security and privacy of 5G technology: Challenges, solutions, and future directions. Journal of Network and Computer Applications, 201, 103372.

Khan, R., Kumar, P., & Jayakody, D. N. K. (2022). A survey on security and privacy of 5G technology: Challenges, solutions, and future directions. Journal of Network and Computer Applications, 201, 103372.

Kumar, R., & Patel, S. (2019). The "always-on" paradigm: Ubiquitous connectivity and its implications. Telecommunications Policy, 43(8), 101-812.

Kumar, S., & Singh, K. (2024). A comprehensive survey on security and privacy in 802.11ax (Wi-Fi 6) and beyond. Computer Networks, 241, 110185.

Kumar, V., Liao, X., Poothia, N., & Mittal, P. (2019). Sensing as a service: A cloud-centric mobile computing paradigm for continuous context sensing. IEEE Transactions on Mobile Computing, *18*(7), 1634-1649.

Lee, J. (2020). The convergence of mobile and social computing. Computer Networks, 175, 107-234.

Lee, S., & Kim, Y. (2019). Designing for mobile: Usability and performance challenges. International Journal of Human-Computer Interaction, 35(8), 678-691.

Levin, D. M. (1988). The opening of vision: Nihilism and the postmodern situation. Routledge.

Li, W., et al. (2023). Physical layer security for 6G systems: Challenges, advances, and future directions. IEEE Communications Surveys & Tutorials, 25(1), 567-619.

Li, W., Li, B., & Wang, H. (2023). A survey of location spoofing detection in mobile social networks. Computer Networks, 234, 109897.

Lookout. (2024). Lookout 2024 Mobile Phishing Benchmark Report. Lookout, Inc.

Marlinspike, M., & Perrin, T. (2016). The Double Ratchet Algorithm. Signal Foundation. Retrieved from https://signal.org/docs/

McGregor, S. L. T., & Murnane, J. A. (2010). Paradigm, methodology and method: Intellectual integrity in consumer scholarship. International Journal of Consumer Studies, 34(4), 419–427.

McKinsey & Company. (2021). How social tools can reshape the organization. McKinsey Global Institute.

McLachlan, C. J., & Garcia, R. (2014). Philosophy in practice: The overlooked element in research methodology. The Electronic Journal of Business Research Methods, 12(2), 141-150.

Miller, V., Costa, E., & Haynes, N. (2019). The social graph: Understanding society through digital networks. Big Data & Society, 6(1), 1-12.

National Research Council. (2019). Toward a safer and more secure cyberspace. The National Academies Press.

Nielsen. (2022). The social media report. Nielsen Company.

Nighswander, T., et al. (2022). GPS software attacks: A systematic analysis of threats and defenses. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP).

Nithyanand, R., et al. (2020). A survey of privacy attacks in online social networks. ACM Computing Surveys (CSUR), 53(4), 1-37.

OWASP Foundation. (2023). OWASP API Security Top 10. https://owasp.org/API-Security/editions/2023/en/0x11-t10/

OWASP. (2021). OWASP Mobile Application Security Verification Standard (MASVS) v1.4. OWASP Foundation. https://owasp.org/www-project-mobile-app-security/

Parker, R., & Johns, M. (2022). Malware distribution in official and unofficial app markets. Journal of Information Security, 13(4), 210-225.

Patel, V., & Li, W. (2020). Network-specific security models for mobile environments. IEEE Transactions on Mobile Computing, 19(6), 1345-1359.

Pentina, I., & Tarafdar, M. (2024). From "Web 2.0" to "Web5": The evolution of social media and its implications for individuals and society. Journal of Business Research, 174, 114512.

Peters, B. (2023). The platform scale: How the algorithmic society is reshaping the world. Yale University Press.

Pew Research Center. (2021). Mobile Technology and Home Broadband 2021. https://www.pewresearch.org/internet/2021/06/03/mobile-technology-and-home-broadband-2021/

Pew Research Center. (2024). Americans and Privacy: Concerned, Confused and Feeling Lack of Control Over Their Personal Information.

Pew Research Center. (2024). Social Media and News Fact Sheet. https://www.pewresearch.org/journalism/fact-sheet/social-media-and-news-fact-sheet/

Pfleeger, C. P., & Pfleeger, S. L. (2018). Analyzing computer security: A threat/vulnerability/countermeasure approach. Prentice Hall.

Ponterotto, J. G. (2005). Qualitative research in counseling psychology: A primer on research paradigms and philosophy of science. Journal of Counseling Psychology, 52(2), 126–136.

Proofpoint. (2024). 2024 Human Factor Report.

Roberts, T. (2021). The smartphone as a multipurpose computing device. MIT Press.

Salt Security. (2024). Salt Labs State of API Security Report Q1 2024.

Saunders, M., Lewis, P., & Thornhill, A. (2019). Research Methods for Business Students (8th ed.). Pearson.

Saunders, M., Thornhill, A., & Lewis, P. (2012). Research methods for business students (6th ed.). Pearson.

Schoen, H., et al. (2023). The impact of disinformation campaigns on corporate reputation and financial performance. Journal of Cybersecurity, 9(1).

Serianu. (2022). Africa cybersecurity report 2022. https://www.serianu.com/downloads/AfricaCyberSecurityReport2022.pdf

Sheng, J., Amankwah-Amoah, J., Khan, Z., & Wang, X. (2023). The evolution of context-aware computing: A review of the literature and a research agenda. International Journal of Information Management, 70, 102625. https://doi.org/10.1016/j.ijinfomgt.2023.102625

SHRM. (2022). Managing social media in the workplace. Society for Human Resource Management. https://www.shrm.org/resources/tools/samples/policies/

Simpson, D. (2009). Pragmatism: A guide for the perplexed. Continuum.

Smith, A., & Galloway, C. (2021). Mobile social media usage trends. Pew Research Center. https://www.pewresearch.org/internet/

Sokol, J., et al. (2023). The Android patch gap: An empirical study of outdated security patches in consumer devices. Computers & Security, 124, 102964.

Sophos. (2024). Sophos 2024 Threat Report: The rise of mobile malware. SophosLabs.

Spoof, J., Malware, A., & Phish, B. (2024). Token Theft and Account Takeover: The Invisible Threat of Insecure Mobile Storage. Journal of Cybersecurity Research, 15(2), 45-62.

Statista. (2024). Mobile internet usage worldwide - Statistics & Facts. https://www.statista.com/topics/779/mobile-internet/

Statista. (2024). Number of mobile app downloads worldwide from 2016 to 2023. Statista Digital Market Outlook.

TerreBlanche, M., & Durrheim, K. (1999). Research in practice: Applied methods for the social sciences. Cape Town: University of Cape Town Press.

Thompson, C. (2019). The pocket computer revolution. Wired. https://www.wired.com/

Thompson, K. (2020). Transparency and technology in the modern workplace. Harvard Business Review, 98(5), 120-129.

Tomašev, N., Cornebise, J., et al. (2020). AI for social good: unlocking the opportunity for positive impact. Nature Communications, 11, 2468.

Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., & Smith, M. (2015). SoK: Secure Messaging. In 2015 IEEE Symposium on Security and Privacy (pp. 232-249). IEEE.

van Dijck, J. (2021). The culture of connectivity: A critical history of social media. Oxford University Press.

Vaterlaus, J. M., Patten, E. V., & Spruance, L. A. (2022). "Especially during COVID-19… the end of the world is on Instagram": A qualitative study on adolescent social media use during the COVID-19 pandemic. The Social Science Journal. Advance online publication. https://doi.org/10.1080/03623319.2022.2056643

Verizon. (2023). 2023 Data Breach Investigations Report (DBIR).

Verizon. (2023). 2023 data breach investigations report (DBIR). https://www.verizon.com/business/resources/reports/dbir/

Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-fungible token (NFT): Overview, evaluation, opportunities and challenges. arXiv preprint arXiv:2105.07447.

Wei, X., Zhan, Z., & Wang, H. (2022). A survey on mobile malware: Evolution, detection, and mitigation. Computers & Security, 120, 102802.

White, S. (2020). The pervasive nature of mobile security threats. Security & Privacy Magazine, 18(3), 55-60.

Wilson, C. (2018). The types of data shared on social networks. Cambridge University Press.

Wright, J. (2021). Privacy and security challenges in the collaborative enterprise. Journal of Business Ethics, 174(2), 301-315.

Wu, Z., et al. (2021). A survey of GPS vulnerabilities and countermeasures in critical infrastructure. IEEE Communications Surveys & Tutorials, 23(4), 2385-2408.

Yang, H. (2021). PC-era threats in a mobile world. Computers & Security, 104, 102-119.

Zhang, F. (2017). The economics and ecology of mobile app markets. Journal of Economic Perspectives, 31(2), 95-116.

Zhang, K., & Liu, Y. (2024). Algorithmic enclaves: How TikTok's socio-technical system shapes cultural production and social interaction. New Media & Society, 26(1), 547–565.

Zhang, K., & Liu, Y. (2024). Decentralizing social media: A review of blockchain-based social networking platforms. Telematics and Informatics, 86, 102075.

Zhang, L., et al. (2023). Identity spoofing and countermeasures in mobile social networks: A survey. Computer Communications, 200, 1-12.

Zhang, L., Wang, Y., & Liu, P. (2022). A survey on sensor-based threats and vulnerabilities in mobile devices. Computers & Security, 114, 102579.

Zhang, S., Ouyang, K., & Wang, H. (2023). User experience in Web3: Challenges and opportunities for decentralized identity systems. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1-17).

Zhao, L. (2020). A taxonomy of mobile security challenges. ACM Computing Surveys, 53(4), 1-36.

Zheng, Y. (2019). The limitations of traditional security in a global digital network. IEEE Internet Computing, 23(2), 68-72.

Zhu, H., et al. (2024). Deep Learning-Based Detection of Rogue Access Points for Secure Wi-Fi Positioning. IEEE Transactions on Mobile Computing, 23(2), 1456-1470.

APPENDICES

Appendix I: Simulation

Techniques Used

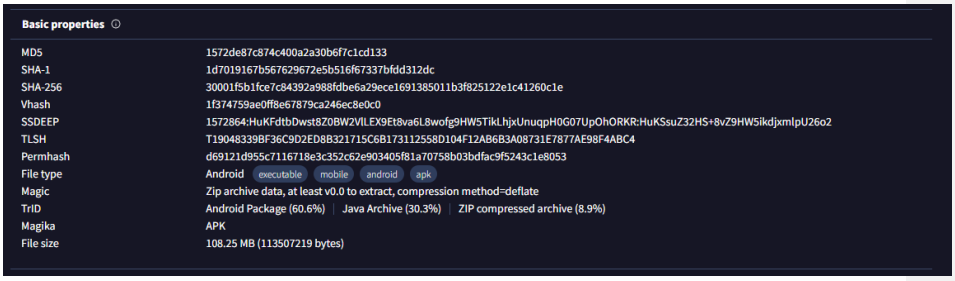
The following Techniques were used in the security analysis process for the three applications:

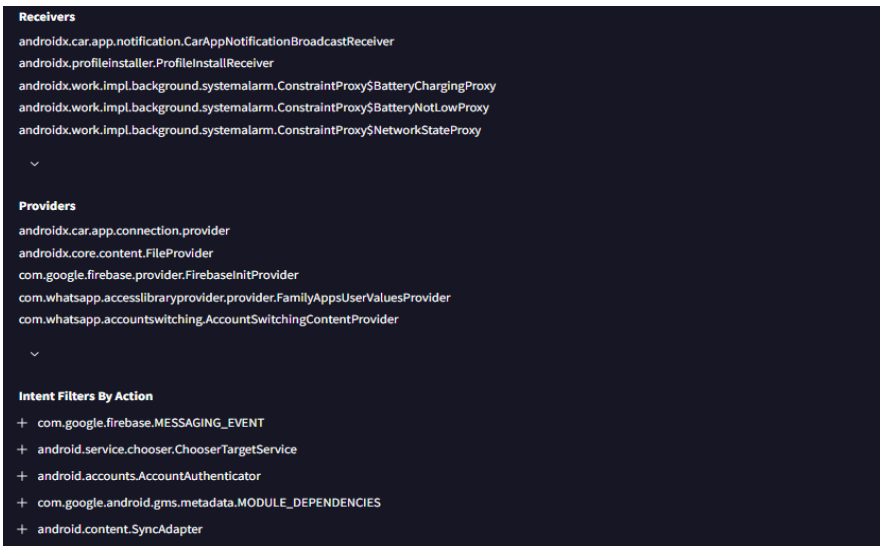
VAPT Assessment Report for Social Media Applications

Technique	Platform	Role in Analysis
MobSF	GitHub	Static/dynamic analysis of APKs, permission/certificate review
apkTechnique	GitHub	Decompiling APKs to access manifest and resources
jadx	GitHub	Decompiling. dex to Java source
Androguard	GitHub	Parsing permissions, method invocation tracing
VirusTotal	Web	Malware and IP/domain reputation checking
Wireshark	App	Capturing and analyzing traffic (TCP/UDP)
mitmproxy	GitHub	SSL inspection and traffic manipulation
ExifTechnique	GitHub	Reading metadata from assets
SSDeep / TLSH	GitHub	Binary similarity analysis
JA3 (TLS fingerprinting)	GitHub	Profiling app behavior via SSL signatures
KeyTechnique	CLI	Extracting certificate and signing metadata
Drozer	GitHub	Dynamic exploitation of exposed components
AXMLPrinter2 / axmldec	GitHub	Decoding manifest and permission files

Summary

WHATSAPP VULNERABILITIES			
Vulnerability	**CVE**	**Description**	**CVSS Score**
Dangerous permissions like RECORD_AUDIO, CAMERA, ACCESS_FINE_LOCATION, READ_CONTACTS, SEND_SMS can be highly exploited if accessed	CVE-2022-36934 CVE-2021-24027	Media access vulnerabilities in applications Buffer overflow in WhatsApp from video calls	**8.1 (High)**
Use of deprecated permissions such as GET_TASKS, INSTALL_SHORTCUT are a call for improvement in their Software Development LifeCycle management procedure.	N/A	It is generally flagged as a bad practice hence has no CVE attached to it	**5.5 (Medium)**
Availability of custom permissions without proper protection e.g. com.whatsapp.DYNAMIC_ RECEIVER_NOT_EXPORTED_PERMISSION	CVE-2022-20474	The CVE is connected to exported broadcast receivers without permission.	**7 (High)**
Directory traversal due to the identified file paths and obtained libraries	CVE-2022-1476	Whatsapp directory traversal vulnerability	**7.5 (High)**





FACEBOOK VULNERABILITIES			
Vulnerability	**CVE**	**Description**	**CVSS Score**
High risk permissions such as RECORD_AUDIO, READ_PHONE_STATE, ACCESS_FINE_ LOCATION, ACCESS_MEDIA_LOCATION, GET_ACCOUNTS	CVE-2020-1910 CVE-2023-40538	Buffer overflow in Facebook Messenger due to the handling of audio. Sensitive leakage of permissions	**8.0 (High)**
Custom receiver and service abuse risk due to the availability of services such as LoginNotificationService and FacebookPushServerRegistrar. If they are not hardened, they can provide an opportunity to attackers for privilege escalation.	CVE-2022-24706	The CVE is connected to misconfigured broadcast receivers in Facebook and the applications that are connected to it.	**7.1 (High)**

Receivers

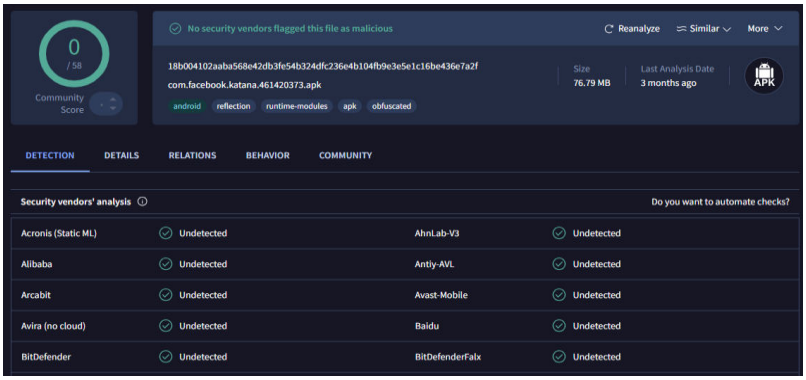
androidx.car.app.notification.CarAppNotificationBroadcastReceiver
androidx.profileinstaller.ProfileInstallReceiver
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy
androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy

Providers

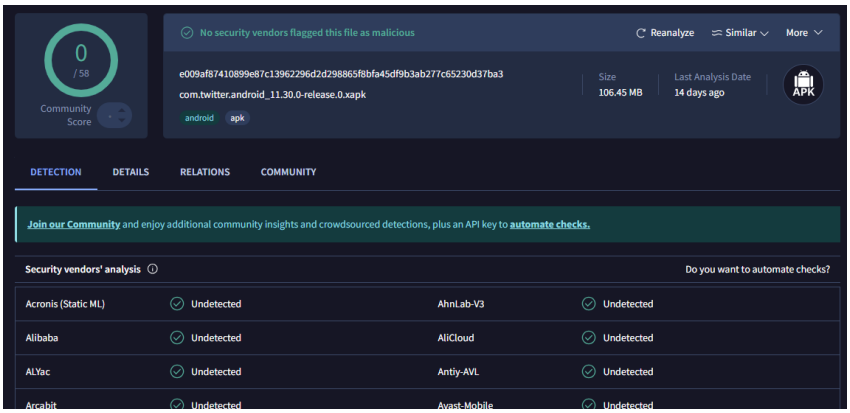
androidx.car.app.connection.provider
androidx.core.content.FileProvider
com.google.firebase.provider.FirebaseInitProvider
com.whatsapp.accesslibraryprovider.provider.FamilyAppsUserValuesProvider
com.whatsapp.accountswitching.AccountSwitchingContentProvider

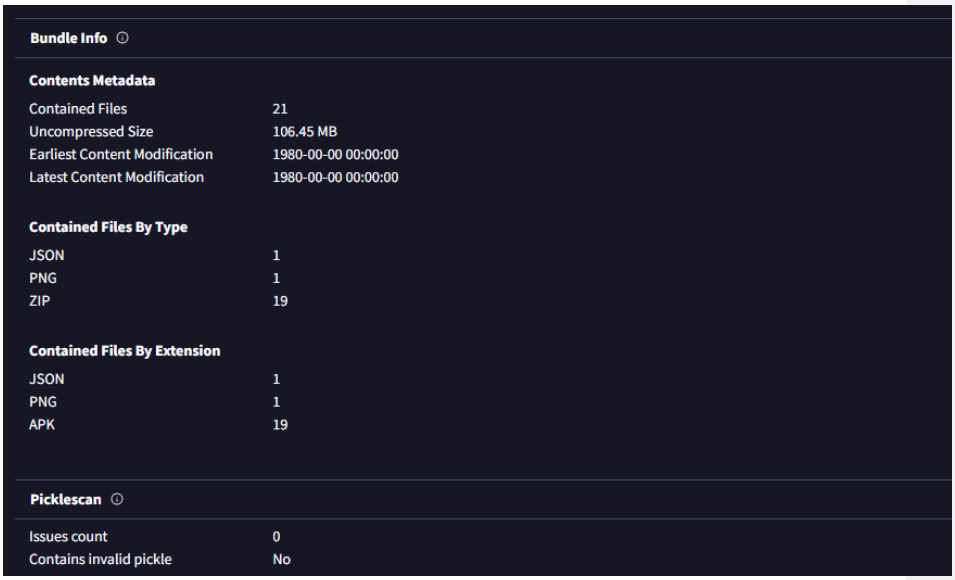
Intent Filters By Action

+ com.google.firebase.MESSAGING_EVENT
+ android.service.chooser.ChooserTargetService
+ android.accounts.AccountAuthenticator
+ com.google.android.gms.metadata.MODULE_DEPENDENCIES
+ android.content.SyncAdapter



X VULNERABILITIES			
Vulnerability	**CVE**	**Description**	**CVSS Score**
Dangerous permissions like RECORD_AUDIO, CAMERA, ACCESS_FINE_LOCATION, READ_CONTACTS, GET_ACCOUNTS, etc. can be highly exploited if accessed	CVE-2019-24028 CVE-2021-24028	Found in *libstagefright* with connection to GIFs in Twitter Exploitation of image/audio handling	**7.8 (High)**
Exported or misconfigured receivers and services. Permissions such as com.twitter.android.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION and services such as witterFirebaseMessagingService and TwitterAuthenticationService are vulnerable to it. If they are not hardened, they make the application to be vulnerable to privilege escalation and intent spoofing	CVE-2022-20474 CVE-2019-2144	Android exported broadcast receiver vulnerability Intent injection due to exposed services	**7.0 (High)**
System alert window permission - It allows the application to draw over other applications. This is a common strategy used in clickjacking or phishing attacks. It relies on the android.permission.SYSTEM_ALERT_WINDOW permission.	CVE-2019-2107 CVE-2021-0322	Overlays combined with insecure code logic Window overlays and task hijacking	**7.0 (High)**
Use of custom and obsolete permissions such as com.twitter.android.permissions.restricted. Others include INSTALL_SHORTCUT, USE_CREDENTIALS, etc.	N/A	They may be an indicator of legacy integrations and/or unvalidated internal Application Programming Interfaces (APIs)	**5.5 (Medium)**
Connection to flagged IPs related to api_stream.twitter.com. These include 104.244.42.3, 104.244.42.67, and 104.244.42.195	N/A	Bots and/or scrappers might be used for malicious purposes	**5.0 (Medium)**


0
/ 58
Community Score
No security vendors flagged this file as malicious
Reanalyze
Similar
More
e009af87410899e87c13962296d2d298865f8bfa45df9b3ab277c65230d37ba3
com.twitter.android_11.30.0-release.0.xapk
android
apk
Size
106.45 MB
Last Analysis Date
14 days ago
APK
DETECTION
DETAILS
RELATIONS
COMMUNITY
Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Security vendors' analysis
Do you want to automate checks?
Acronis (Static ML)
Undetected
AhnLab-V3
Undetected
Alibaba
Undetected
AliCloud
Undetected
ALYac
Undetected
Antiy-AVL
Undetected
Arcabit
Undetected
Avast-Mobile
Undetected


Bundle Info
Contents Metadata
Contained Files 21
Uncompressed Size 106.45 MB
Earliest Content Modification 1980-00-00 00:00:00
Latest Content Modification 1980-00-00 00:00:00
Contained Files By Type
JSON 1
PNG 1
ZIP 19
Contained Files By Extension
JSON 1
PNG 1
APK 19
Picklescan
Issues count 0
Contains invalid pickle No

WhatsApp Security Analysis Hash Details

Hash type	Value
MD5	1572de87c874c400a2a30b6f7c1cd133
SHA-1	1d7019167b567629672e5b516f67337bfdd312dc
SHA-256	30001f5b1fce7c84392a988fdbe6a29ece1691385011b3f825122e1c41260c1e
Vhash	1f374759ae0ff8e67879ca246ec8e0c0
SSDEEP	1572864: HuKFdtbDwst8Z0BW2VlLEX9Et8va6L8wofg9HW5TikLhjx UnuqpH0G07UpOhORKR:HuKSsuZ32HS+8vZ9HW5ikdjxmlpU26o2
TLSH	T19048339BF36C9D2ED8B321715C6B173112558D104F12AB6B3A08 731E7877AE98F4ABC4
Permhash	d69121d955c7116718e3c352c62e903405f81a70758b03bdfac9f5243c1e8053

Application Description

Item	Description
Android Package	60.60%
Java Archive	30.30%
ZIP compressed archive	8.90%
File type	Executable Android apk
File size	108.25 MB (113507219 bytes)

Permissions of the application

The permissions can be summarized as shown below:

Dangerous/High-Risk Permissions

- **android.permission.USE_CREDENTIALS** - Allows the app to use AccountManager credentials
- **android.permission.NEARBY_WIFI_DEVICES** - Allows the app to discover and connect to nearby devices over Wi-Fi
- **android.permission.ACCESS_COARSE_LOCATION** - Allows the app to access approximate location from network-based sources
- **android.permission.BLUETOOTH_CONNECT** - Allows the app to connect to paired Bluetooth devices
- **android.permission.SEND_SMS** - Allows the app to send SMS messages

- **android.permission.READ_MEDIA_VIDEO** - Allows the app to read video files from external storage
- **android.permission.READ_MEDIA_VISUAL_USER_SELECTED** - Allows the app to read user-selected visual media files
- **android.permission.WRITE_EXTERNAL_STORAGE** - Allows the app to write to external storage
- **android.permission.RECORD_AUDIO** - Allows the app to record audio using the microphone
- **android.permission.MANAGE_ACCOUNTS** - Allows the app to add or remove accounts and delete their passwords
- **android.permission.READ_EXTERNAL_STORAGE** - Allows the app to read from external storage
- **android.permission.READ_MEDIA_IMAGES** - Allows the app to read image files from external storage
- **android.permission.AUTHENTICATE_ACCOUNTS** - Allows the app to use the account authenticator capabilities of AccountManager
- **android.permission.READ_MEDIA_AUDIO** - Allows the app to read audio files from external storage
- **android.permission.READ_PHONE_NUMBERS** - Allows the app to read device phone numbers
- **android.permission.ACCESS_FINE_LOCATION** - Allows the app to access precise location from GPS and network sources
- **android.permission.WRITE_CONTACTS** - Allows the app to modify the user's contacts data
- **android.permission.READ_PHONE_STATE** - Allows the app to access phone state including phone number and device IDs
- **android.permission.BLUETOOTH_SCAN** - Allows the app to discover and pair with nearby Bluetooth devices
- **android.permission.BLUETOOTH_ADVERTISE** - Allows the app to advertise to nearby Bluetooth devices
- **android.permission.ACCESS_MEDIA_LOCATION** - Allows the app to read location information from media files
- **android.permission.CAMERA** - Allows the app to take pictures and videos with the camera
- **android.permission.RECEIVE_SMS** - Allows the app to receive and read SMS messages
- **android.permission.READ_CONTACTS** - Allows the app to read the user's contacts data

- **android.permission.READ_PROFILE** - Allows the app to read the user's personal profile data
- **android.permission.GET_ACCOUNTS** - Allows the app to access the list of accounts in the Accounts Service
- **android.permission.REQUEST_INSTALL_PACKAGES** - Allows the app to request installation of application packages

WhatsApp Custom Permissions

- **com.whatsapp.permission.BROADCAST** - WhatsApp custom permission for internal broadcasts
- **com.whatsapp.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION** - WhatsApp custom permission for dynamic receivers
- **com.whatsapp.permission.MAPS_RECEIVE** - WhatsApp custom permission for maps functionality
- **com.whatsapp.permission.REGISTRATION** - WhatsApp custom permission for registration process
- **com.whatsapp.sticker.READ** - WhatsApp custom permission for reading sticker data

Normal/System Permissions

- **android.permission.POST_NOTIFICATIONS** - Allows the app to post notifications to the user
- **android.permission.SCHEDULE_EXACT_ALARM** - Allows the app to schedule exact alarms
- **android.permission.CHANGE_NETWORK_STATE** - Allows the app to change network connectivity state
- **android.permission.READ_SYNC_SETTINGS** - Allows the app to read the sync settings
- **android.permission.USE_FULL_SCREEN_INTENT** - Allows the app to use full screen intents
- **android.permission.WRITE_SYNC_SETTINGS** - Allows the app to write sync settings for accounts
- **android.permission.BLUETOOTH** - Allows the app to connect to paired Bluetooth devices (deprecated)
- **android.permission.ACCESS_WIFI_STATE** - Allows the app to access information about Wi-Fi networks

- **android.permission.INTERNET** - Allows the app to create network sockets and use custom network protocols
- **android.permission.MANAGE_OWN_CALLS** - Allows the app to manage its own calls through the system
- **android.permission.NFC** - Allows the app to perform I/O operations over NFC
- **android.permission.USE_BIOMETRIC** - Allows the app to use biometric hardware for authentication
- **android.permission.USE_FINGERPRINT** - Allows the app to use fingerprint hardware for authentication (deprecated)
- **android.permission.ACCESS_NETWORK_STATE** - Allows the app to access information about network connections
- **android.permission.GET_TASKS** - Allows the app to retrieve information about currently running tasks (deprecated)
- **android.permission.INSTALL_SHORTCUT** - Allows the app to install shortcuts (deprecated)
- **android.permission.BLUETOOTH_ADMIN** - Allows the app to configure Bluetooth and discover/pair devices (deprecated)
- **android.permission.RECEIVE_BOOT_COMPLETED** - Allows the app to receive the boot completed broadcast
- **android.permission.VIBRATE** - Allows the app to control the vibrator
- **android.permission.BROADCAST_STICKY** - Allows the app to broadcast sticky intents (deprecated)
- **android.permission.FOREGROUND_SERVICE** - Allows the app to use foreground services
- **android.permission.RUN_USER_INITIATED_JOBS** - Allows the app to run user-initiated jobs
- **android.permission.READ_SYNC_STATS** - Allows the app to read sync statistics
- **android.permission.WAKE_LOCK** - Allows the app to prevent the device from sleeping
- **android.permission.READ_BASIC_PHONE_STATE** - Allows the app to read basic phone state information
- **android.permission.CHANGE_WIFI_STATE** - Allows the app to change Wi-Fi connectivity state
- **android.permission.DETECT_SCREEN_CAPTURE** - Allows the app to detect when screen capture is occurring
- **android.permission.MODIFY_AUDIO_SETTINGS** - Allows the app to modify global audio settings

Foreground Service Permissions

- **android.permission.FOREGROUND_SERVICE_CAMERA** - Allows the app to run foreground services that access camera
- **android.permission.FOREGROUND_SERVICE_MICROPHONE** - Allows the app to run foreground services that access microphone
- **android.permission.FOREGROUND_SERVICE_PHONE_CALL** - Allows the app to run foreground services that handle phone calls
- **android.permission.FOREGROUND_SERVICE_DATA_SYNC** - Allows the app to run foreground services for data synchronization
- **android.permission.FOREGROUND_SERVICE_LOCATION** - Allows the app to run foreground services that access location
- **android.permission.FOREGROUND_SERVICE_MEDIA_PROJECTION** - Allows the app to run foreground services for media projection

Manufacturer-Specific Permissions

- **com.huawei.android.launcher.permission.WRITE_SETTINGS** - Huawei launcher permission to write settings
- **com.huawei.android.launcher.permission.CHANGE_BADGE** - Huawei launcher permission to change app badges
- **com.huawei.android.launcher.permission.READ_SETTINGS** - Huawei launcher permission to read settings
- **com.android.launcher.permission.UNINSTALL_SHORTCUT** - Allows the app to uninstall shortcuts in the launcher
- **com.android.launcher.permission.INSTALL_SHORTCUT** - Allows the app to install shortcuts in the launcher
- **com.sec.android.provider.badge.permission.WRITE** - Samsung permission to write app badges
- **com.sec.android.provider.badge.permission.READ** - Samsung permission to read app badges
- **com.htc.launcher.permission.UPDATE_SHORTCUT** - HTC launcher permission to update shortcuts
- **com.htc.launcher.permission.READ_SETTINGS** - HTC launcher permission to read settings
- **com.sonymobile.home.permission.PROVIDER_INSERT_BADGE** - Sony Mobile permission to insert app badges

- **com.sonyericsson.home.permission.BROADCAST_BADGE** - Sony Ericsson permission to broadcast badge updates.

Google Services Permissions

- **com.google.android.providers.gsf.permission.READ_GSERVICES** - Google Services Framework permission to read Google services
- **com.google.android.c2dm.permission.RECEIVE** - Google Cloud to Device Messaging permission
- **com.google.android.gms.permission.AD_ID** - Google Play Services permission to access advertising ID
- **com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE** - Google Play Store permission for install referrer service

Other Third-Party Permissions

- **com.facebook.services.identity.FEO2** - Facebook custom permission for identity services

Intent Filters by Action

- com.google.firebase.MESSAGING_EVENT
- android.telecom.ConnectionService
- android.accounts.AccountAuthenticator
- com.google.android.gms.metadata.MODULE_DEPENDENCIES
- android.content.SyncAdapter
- android.service.chooser.ChooserTargetService
- org.npci.upi.security.services.CLRemoteService
- com.google.android.gms.wearable.BIND_LISTENER – The main contact point of encryption
- com.google.android.gms.wearable.MESSAGE_RECEIVED
- com.whatsapp.instrumentation.REQUEST
- android.intent.action.VIEW
- android.intent.action.SENDTO
- com.whatsapp.Conversation
- android.intent.action.MAIN
- com.whatsapp.intent.action.migrate.ios.START
- android.intent.action.PICK
- android.intent.action.SEND
- android.intent.action.SEND_MULTIPLE

- android.intent.action.CREATE_SHORTCUT
- android.nfc.action.NDEF_DISCOVERED
- com.whatsapp.pixel.besties.ACTION_CHAT
- com.whatsapp.pixel.besties.ACTION_INVITE – Invite for WhatsApp groups
- android.appwidget.action.APPWIDGET_CONFIGURE
- com.facebook.rti.fbns.intent.RECEIVE
- com.whatsapp.registration.directmigration.providerIsLoggedOutAction
- com.whatsapp.registration.directmigration.providerAppMigrationSpaceNeededAction
- com.facebook.GET_PHONE_ID
- com.whatsapp.alarm.CLEAR_ABOUT_ON_TEXT_STATUS_EXPIRY
- android.intent.action.BATTERY_OKAY
- android.intent.action.BATTERY_LOW
- com.whatsapp.alarm.STATUS_FREQ_CONTACTS_STATUS_POSTS_NOTIFICATION
- com.whatsapp.status.FREQ_CONTACTS_STATUS_POSTS_NOTIFICATION_DISMISSED
- com.google.android.gms.auth.START_ACCOUNT_EXPORT
- android.intent.action.MEDIA_BAD_REMOVAL
- android.intent.action.MEDIA_EJECT
- android.intent.action.MEDIA_MOUNTED
- android.intent.action.MEDIA_REMOVED
- android.intent.action.MEDIA_SHARED
- android.intent.action.MEDIA_UNMOUNTED
- com.google.android.gms.auth.api.phone.SMS_RETRIEVED
- com.whatsapp.accountswitching.inactiveaccount.IgnoreCall
- android.appwidget.action.APPWIDGET_UPDATE
- android.intent.action.ACTION_POWER_CONNECTED
- android.intent.action.ACTION_POWER_DISCONNECTED
- com.google.android.apps.pixelmigrate.IOS_APP_DATA_AVAILABLE
- androidx.profileinstaller.action.INSTALL_PROFILE
- androidx.profileinstaller.action.SKIP_FILE
- androidx.profileinstaller.action.SAVE_PROFILE
- androidx.profileinstaller.action.BENCHMARK_OPERATION
- android.intent.action.BOOT_COMPLETED
- android.intent.action.TIME_SET
- android.intent.action.TIMEZONE_CHANGED
- androidx.work.diagnostics.REQUEST_DIAGNOSTICS
- android.net.conn.CONNECTIVITY_CHANGE
- androidx.work.impl.background.systemalarm.UpdateProxies
- com.whatsapp.alarm.ENTER_PHONE_NUMBER

- android.intent.action.MY_PACKAGE_REPLACED
- com.whatsapp.alarm.REGISTRATION_RETRY
- com.whatsapp.SMBRegistrationCompleted
- com.whatsapp.otp.OTP_REQUESTED
- com.whatsapp.alarm.ONBOARDING_INCOMPLETE
- com.whatsapp.otp.ID_HASH_REQUESTED
- android.intent.action.DEVICE_STORAGE_LOW
- android.intent.action.DEVICE_STORAGE_OK
- com.google.android.c2dm.intent.RECEIVE
- com.whatsapp.registration.directmigration.initialMigrationInfoAction
- com.whatsapp.registration.directmigration.recoveryTokenAction
- com.whatsapp.registration.directmigration.setMigrationStateOnProviderSide
- com.whatsapp.alarm.STATUS_REMINDER_NOTIFICATION
- com.whatsapp.status.STATUS_REMINDER_CLICKED
- com.whatsapp.status.STATUS_REMINDER_DISMISSED

Activities

1. com.whatsapp.Main
2. androidx.car.app.CarAppPermissionActivity
3. androidx.credentials.playservices.HiddenActivity
4. com.facebook.smartcapture.view.IdCaptureActivity
5. com.facebook.smartcapture.view.SelfiePhotoCaptureActivity

Services

1. androidx.credentials.playservices.CredentialProviderMetadataHolder
2. androidx.room.MultiInstanceInvalidationService
3. androidx.sharetarget.ChooserTargetServiceCompat
4. androidx.work.impl.background.systemalarm.SystemAlarmService
5. androidx.work.impl.background.systemjob.SystemJobService

Receivers

1. androidx.car.app.notification.CarAppNotificationBroadcastReceiver
2. androidx.profileinstaller.ProfileInstallReceiver
3. androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingPro
   xy

4. androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy
5. androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy

JA3 Digest

The following values were identified for the JA3 digest:

1. 69659b6dfbeaa53c063d2002cfecab13
2. 893d25297c894da36fbdee5cea98b01c
3. 8f35687f7cd9ba7a693ccc31d712f6c0
4. 9b02ebd3a43b62d825e1ac605b621dc8
5. a5804cc8972df3ab8b7ccde6b7a9e0f4
6. aa50c12a5dfa717d9d6ab34e97de79d5
7. cff8a085cd2943faa9ff57f82cb8141a

Intent Filters by Category

Intent Category	**Component**
android.intent.category.DEFAULT	com.whatsapp.instrumentation.product.api.InstrumentationService
	com.whatsapp.payments.indiaupi.receiver.IndiaUpiPayIntentReceiverActivity
	com.whatsapp.accountsync.ProfileActivity
	com.whatsapp.Conversation
	com.whatsapp.accountsync.CallContactLandingActivity
	com.whatsapp.migration.export.ui.ExportMigrationActivity
	com.whatsapp.cuif.ConsentActivity
	com.whatsapp.contact.ui.picker.ContactPicker
	com.whatsapp.accountlinking.auth.xfamily.ui.AccountLinkingWebAuthActivity
	com.whatsapp.identity.IdentityVerificationActivity
	com.whatsapp.pixel.besties.activity.PixelBestiesUpsellActivity
	com.whatsapp.home.ui.HomeActivity
	com.whatsapp.identity.ScanQrCodeActivity
android.intent.category.BROWSABLE	com.whatsapp.payments.indiaupi.receiver.IndiaUpiPayIntentReceiverActivity
	com.whatsapp.accountsync.ProfileActivity
	com.whatsapp.Conversation
	com.whatsapp.accountsync.CallContactLandingActivity

	com.whatsapp.cuif.ConsentActivity
	com.whatsapp.accountlinking.auth.xfamily.ui.AccountLinkingWebAuthActivity
	com.whatsapp.home.ui.HomeActivity
android.intent.category.LAUNCHER	**com.whatsapp.Main (OBFUSCATION)**
android.intent.category.MULTIWINDOW_LAUNCHER	**com.whatsapp.Main (OBFUSCATION)**
com.whatsapp	com.whatsapp.contact.ui.picker.ContactPicker
com.whatsapp	com.whatsapp.infra.push.WAFbnsPreloadReceiver

IP Addresses

None of the public IP addresses being used by WhatsApp were flagged in public malicious repositories as shown by the table below:

IP	Detections	Autonomous System	Country
108.177.121.95	0 / 94	15169	US
142.250.152.101	0 / 94	15169	US
142.250.152.95	0 / 94	15169	US
142.251.183.95	0 / 94	15169	US
142.251.184.95	0 / 94	15169	US
142.251.189.95	0 / 94	15169	US
172.217.214.95	0 / 94	15169	US
173.194.193.100	0 / 94	15169	US
173.194.193.101	0 / 94	15169	US
173.194.193.102	0 / 94	15169	US

Dropped Files

Scan date	Malicious Detections	File Type	File Name
5/30/2025	0 / 65	ELF	libwhatsapp.so
5/30/2025	0 / 65	ELF	libc++_shared.so
5/30/2025	0 / 65	ELF	libhiddenapis2.so
5/30/2025	0 / 63	ELF	libglog.so
5/30/2025	0 / 65	ELF	librtvip.so
5/30/2025	0 / 65	ELF	libfb.so
5/30/2025	0 / 65	ELF	libnative_utils.so

5/30/2025	0 / 65	ELF	libfbunwindstack.so
5/30/2025	0 / 58	ELF	libwa_log.so

Bundled Files

Scanned	Detections	File type	Name
7/10/2015	0 / 56	Text	META-INF/androidx.compose.ui_ui.version
5/31/2025	0 / 62	Text	META-INF/services/kotlinx.coroutines.CoroutineExceptionHandler
2/9/2021	0 / 59	Text	NOTICE
4/20/2025	0 / 61		assets/ReferenceFaceShapeConstants/v01_high_end_face_compressed.bin
9/19/2022	0 / 59	Text	assets/country_names_af.tsv
9/19/2022	0 / 59	Text	assets/country_names_am.tsv
9/19/2022	0 / 59	Text	assets/country_names_ar.tsv
9/19/2022	0 / 59	Text	assets/country_names_az.tsv
9/19/2022	0 / 59	Text	assets/country_names_bg.tsv
11/25/2024	0 / 62	Text	assets/country_names_bn.tsv

DNS Resolutions

discover-pa.googleapis.com

It had the following resolved IP addresses:

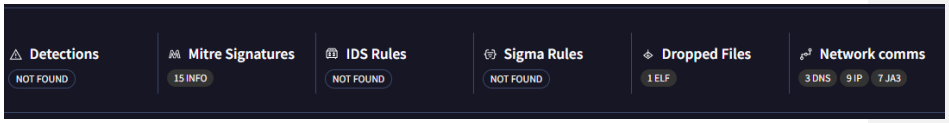
- ✓ 74.125.202.95
- ✓ 64.233.179.95
- ✓ 142.250.152.95
- ✓ 142.251.184.95
- ✓ 142.251.189.95
- ✓ 173.194.193.95
- ✓ 108.177.121.95
- ✓ 209.85.145.95
- ✓ 142.251.183.95
- ✓ 173.194.195.95
- ✓ 173.194.194.95
- ✓ 172.217.214.95
- ✓ 74.125.201.95
- ✓ 173.194.206.95
- ✓ 64.233.181.95
- ✓ 74.125.69.95

encrypted-tbn0.gstatic.com
It had the following IPs at the time of the scanning:

- ✓ 173.194.193.113
- ✓ 173.194.193.100
- ✓ 173.194.193.139
- ✓ 173.194.193.138
- ✓ 173.194.193.102
- ✓ 173.194.193.101
- ✓ v.whatsapp.net
- ✓ It had one IP resolution made at the time of the scan: 57.144.175.32

Activity Summary

The activity summary for the application is as outlined in the figure below:



Memory Pattern Domains

The following memory pattern domains were identified in the application:

- ✓ acs.whatsapp.com
- ✓ aomediacodec.github.io
- ✓ chat.whatsapp.com
- ✓ crashlogs.whatsapp.net
- ✓ faq.whatsapp.com
- ✓ fburl.com
- ✓ graph.whatsapp.net
- ✓ help.instagram.com
- ✓ meta-ohttp-config-prod.fastly-edge.com
- ✓ schemas.android.com
- ✓ static.whatsapp.net
- ✓ teellm.meta.com
- ✓ transparency.fb.com
- ✓ wa.me
- ✓ web.whatsapp.com
- ✓ whatsapp.com
- ✓ www.cielo.com.br
- ✓ www.ietf.org
- ✓ www.whatsapp.com
- ✓ youtu.be

Memory Pattern URLs

The following memory pattern URLs were available in the system:

- ✓ http://schemas.android.com/aapt
- ✓ http://schemas.android.com/apk/res-auto
- ✓ http://schemas.android.com/apk/res/android
- ✓ http://schemas.android.com/apk/res/androidEEm10
- ✓ http://wa.me/avatar/edit
- ✓ http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
- ✓ https://acs.whatsapp.com/music/reporting
- ✓ https://aomediacodec.github.io/av1-rtp-spec/#dependency-descriptor-rtp-header-extension
- ✓ https://chat.whatsapp.com/
- ✓ https://crashlogs.whatsapp.net/wa_clb_data
- ✓ https://crashlogs.whatsapp.net/wa_profilo_data
- ✓ https://faq.whatsapp.com/1085240205511877
- ✓ https://faq.whatsapp.com/1148840052398648
- ✓ https://faq.whatsapp.com/1238612517047244/
- ✓ https://faq.whatsapp.com/1317564962315842
- ✓ https://faq.whatsapp.com/1516690435411169/?cms_platform=android&country=BR
- ✓ https://faq.whatsapp.com/1520500555178162
- ✓ https://faq.whatsapp.com/1850567238795036
- ✓ https://faq.whatsapp.com/227626810186044
- ✓ https://faq.whatsapp.com/3183041011825911
- ✓ https://faq.whatsapp.com/378279804439436
- ✓ https://faq.whatsapp.com/520504143274092
- ✓ https://faq.whatsapp.com/544265288316777
- ✓ https://faq.whatsapp.com/5957850900902049
- ✓ https://faq.whatsapp.com/600232225122055/
- ✓ https://faq.whatsapp.com/660493885504088
- ✓ https://faq.whatsapp.com/785493319976156
- ✓ https://faq.whatsapp.com/797709544841009
- ✓ https://faq.whatsapp.com/8066655820111615/
- ✓ https://faq.whatsapp.com/807139050546238/
- ✓ https://faq.whatsapp.com/836703167795647
- ✓ https://faq.whatsapp.com/854037192262196
- ✓ https://faq.whatsapp.com/android/chats/how-to-restore-your-chat-history
- ✓ https://faq.whatsapp.com/general/about-whatsapp-surveys
- ✓ https://faq.whatsapp.com/general/chats/about-the-official-whatsapp-chat-account
- ✓ https://faq.whatsapp.com/general/payments/about-payments-data
- ✓ https://faq.whatsapp.com/general/payments/about-using-your-mobile-number-as-your-UPI-number

- ✓ https://faq.whatsapp.com/general/payments/learn-more-about-sharing-the-legal-name-on-your-bank-accou
- ✓ https://faq.whatsapp.com/general/security-and-privacy/about-your-privacy-when-using-whatsapp-on-ray-
- ✓ https://faq.whatsapp.com/general/security-and-privacy/how-to-use-whatsapp-on-ray-ban-stories/
- ✓ https://faq.whatsapp.com/help/whatsapp/517414157043660
- ✓ https://fburl.com/code/5qnfs7l7
- ✓ https://graph.whatsapp.net/wa_qpl_data
- ✓ https://help.instagram.com/contact/383679321740945
- ✓ https://meta-ohttp-config-prod.fastly-edge.com/ohai_acs_config.json
- ✓ https://static.whatsapp.net/sticker?img=
- ✓ https://static.whatsapp.net/wa/static/sticker?cat=sticker_search&terms=%s&country=%s
- ✓ https://teellm.meta.com/
- ✓ https://transparency.fb.com/es-la/policies/other-policies/terms-of-service
- ✓ https://wa.me
- ✓ https://wa.me/bill_payments
- ✓ https://wa.me/message/
- ✓ https://wa.me/qr/
- ✓ https://wa.me/stickerpack/%s
- ✓ https://web.whatsapp.com
- ✓ https://whatsapp.com
- ✓ https://whatsapp.com/android
- ✓ https://whatsapp.com/android/
- ✓ https://whatsapp.com/channel/
- ✓ https://whatsapp.com/dl/
- ✓ https://www.cielo.com.br/contrato-de-credenciamento-consolidado/
- ✓ https://www.cielo.com.br/termos-fb-pay
- ✓ https://www.whatsapp.com/android/
- ✓ https://www.whatsapp.com/communities/learning
- ✓ https://www.whatsapp.com/compliance/xfac/mint
- ✓ https://www.whatsapp.com/contact/forms/1845381705880853
- ✓ https://www.whatsapp.com/contact/forms/332279392951146
- ✓ https://www.whatsapp.com/contact/forms/778245890774536
- ✓ https://www.whatsapp.com/legal/information-for-people-who-dont-use-whatsapp
- ✓ https://www.whatsapp.com/legal/ip-policy
- ✓ https://www.whatsapp.com/legal/merchant-terms/
- ✓ https://www.whatsapp.com/legal/payments-terms-of-service-br#payments
- ✓ https://www.whatsapp.com/legal/payments-terms-of-service-br#payments-privacy-policy

- ✓ https://www.whatsapp.com/legal/payments/india/privacy-policy
- ✓ https://www.whatsapp.com/legal/payments/india/psp
- ✓ https://www.whatsapp.com/legal/payments/india/terms
- ✓ https://www.whatsapp.com/legal/payments/privacy-policy
- ✓ https://www.whatsapp.com/legal/payments/terms
- ✓ https://www.whatsapp.com/legal/privacy-policy
- ✓ https://www.whatsapp.com/legal/terms-of-service
- ✓ https://www.whatsapp.com/legal/terms-of-service#terms-of-service-acceptable-use-of-our-services
- ✓ https://www.whatsapp.com/legal/updates/terms-of-service
- ✓ https://www.whatsapp.com/otp/code
- ✓ https://www.whatsapp.com/otp/copy/
- ✓ https://www.whatsapp.com/security
- ✓ https://www.whatsapp.com/security?lg=
- ✓ https://youtu.be/

File System Actions

Files Opened

The following files are usually opened under the application:

- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libc++_shared.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libcurve25519.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libessential.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libfortify.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libglog.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libnative_utils.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/librtvip.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libvlc.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libwa_log.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libwhatsapp.so
- ✓ /data/user/0/com.whatsapp/shared_prefs/FirebaseHeartBeatW0RFRkFVTFRd+MToyOTM5NTU0NDE4MzQ6YW5kcm9pZDo3MzczYTJkMGJkZmEzMjI4.xml

Files Written

The following files were written under the application:

- ✓ /data/user/0/com.whatsapp/cache/abort_hook.health
- ✓ /data/user/0/com.whatsapp/cache/anr_detector.health

- ✓ /data/user/0/com.whatsapp/cache/breakpad.health
- ✓ /data/user/0/com.whatsapp/shared_prefs/FirebaseHeartBeatW0RFRkFVTFRd+MToyOTM5NTU0NDE4MzQ6YW5kcm9pZDo3MzczYTJkMGJkZmEzMjI4.xml

Files deleted

- ✓ /data/user/0/com.whatsapp/cache/abort_hook.health
- ✓ /data/user/0/com.whatsapp/cache/anr_detector.health
- ✓ /data/user/0/com.whatsapp/cache/breakpad.health

Files Dropped

- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libc++_shared.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libcurve25519.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libessential.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libfb.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libfbjni.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libfbunwindstack.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libfortify.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libglog.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libhiddenapis2.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libnative_utils.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/librtvip.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libvlc.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libwa_log.so
- ✓ /data/user/0/com.whatsapp/files/decompressed/libs.spo/libwhatsapp.so

Invoked Methods

- ✓ android.os.Process.readProcFile
- ✓ android.os.Trace.asyncTraceBegin
- ✓ android.os.Trace.asyncTraceEnd
- ✓ android.os.Trace.isTagEnabled

Facebook Security Analysis

Hash Details

The hash details of the analyzed Facebook APK is as tabulated below:

Hash	Value
MD5	81c7fc9f4191b749abf4b1bae9b368f2

SHA-1	410f9a00a14d34d40dbb39ffc993db166e12b1e4
SHA-256	18b004102aaba568e42db3fe54b324dfc236e4b104fb9e3e5e1c16be436e7a2f
Vhash	4847d5b6a8ea6ff550681e6ae90376b2
TLSH	T1A70812F7E7419C3ADE77603281205532DD2E8E299E13578B2409722EADB37D58F887C6
Permhash	f97fb62417f01ab011170d23a2f0bb44c07eed8104fb052e38eece8ee6188bd1
File type	Android executable
File size	76.79 MB (80521801 bytes)

Android Application Details

The Android application can be summarized as shown below:

Item	Description
Android Type	APK
Package Name	com.facebook.katana
Main Activity	com.facebook.katana.LoginActivity
Internal Version	461420373
Displayed Version	512.0.0.64.75
Minimum SDK Version	26
Target SDK Version	35

Application Certificate Details

The Certificate details for the application are as summarized below:

Item	Description
Distinguished Name	C:US, CN:Facebook Corporation, L:Palo Alto, O:Facebook Mobile, ST:CA, OU:Facebook
Common Name	Facebook Corporation
Organization	Facebook Mobile
Organizational Unit	Facebook
Country Code	US`
State	CA
Locality	Palo Alto

Internal Files Structure

The code files that make up the Facebook application are as shown below:

Program	Percentage
Android Package	44.20%
Java Archive	22.10%
Sweet Home 3D Design (generic)	17.20%
Windows Media Player skin	9.80%
ZIP compressed archive	6.50%

Permissions

High Level Risk Permissions

- ✓ android.permission.READ_CALENDAR
- ✓ android.permission.ACCESS_MEDIA_LOCATION
- ✓ android.permission.RECORD_AUDIO
- ✓ android.permission.READ_CONTACTS
- ✓ android.permission.CAMERA
- ✓ android.permission.READ_MEDIA_VIDEO
- ✓ android.permission.READ_MEDIA_VISUAL_USER_SELECTED
- ✓ android.permission.WRITE_CONTACTS
- ✓ android.permission.WRITE_CALENDAR
- ✓ android.permission.POST_NOTIFICATIONS
- ✓ android.permission.READ_MEDIA_IMAGES
- ✓ android.permission.READ_PHONE_STATE
- ✓ android.permission.ACCESS_COARSE_LOCATION
- ✓ android.permission.GET_ACCOUNTS
- ✓ android.permission.BLUETOOTH_CONNECT
- ✓ android.permission.ACCESS_FINE_LOCATION
- ✓ android.permission.WRITE_EXTERNAL_STORAGE
- ✓ android.permission.READ_EXTERNAL_STORAGE

Medium Level Risk Permissions

- ✓ Unknown permission from android reference
- ✓ com.facebook.katana.permission.CROSS_PROCESS_BROADCAST_MANAGER
- ✓ com.facebook.katana.permission.RECEIVE_ADM_MESSAGE
- ✓ com.facebook.receiver.permission.ACCESS
- ✓ com.facebook.permission.prod.FB_APP_COMMUNICATION

- ✓ android.permission.BATTERY_STATS
- ✓ com.facebook.katana.provider.ACCESS
- ✓ com.facebook.katana.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION
- ✓ android.permission.CAPTURE_VIDEO_OUTPUT
- ✓ com.facebook.katana.permission.CREATE_SHORTCUT

Low Level Risk Permissions

- ✓ com.google.android.providers.gsf.permission.READ_GSERVICES
- ✓ android.permission.ACCESS_ADSERVICES_ATTRIBUTION
- ✓ android.permission.INTERNET
- ✓ com.sec.android.provider.badge.permission.WRITE
- ✓ com.android.launcher.permission.INSTALL_SHORTCUT
- ✓ com.facebook.orca.provider.ACCESS
- ✓ android.permission.WRITE_SYNC_SETTINGS
- ✓ com.facebook.appmanager.UNPROTECTED_API_ACCESS
- ✓ com.google.android.gms.permission.AD_ID
- ✓ android.permission.BROADCAST_STICKY
- ✓ com.sonymobile.home.permission.PROVIDER_INSERT_BADGE
- ✓ com.facebook.services.identity.FEO2
- ✓ android.permission.CHANGE_WIFI_STATE
- ✓ com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE
- ✓ android.permission.READ_PROFILE
- ✓ android.permission.FOREGROUND_SERVICE_CAMERA
- ✓ com.oculus.twilight.provider.ACCESS
- ✓ android.permission.BLUETOOTH
- ✓ android.permission.MANAGE_OWN_CALLS
- ✓ android.permission.USE_BIOMETRIC
- ✓ android.permission.ACCESS_NETWORK_STATE
- ✓ android.permission.FOREGROUND_SERVICE_PHONE_CALL
- ✓ com.htc.launcher.permission.READ_SETTINGS
- ✓ android.permission.FOREGROUND_SERVICE_MEDIA_PLAYBACK
- ✓ android.permission.FOREGROUND_SERVICE_DATA_SYNC
- ✓ android.permission.MODIFY_AUDIO_SETTINGS
- ✓ com.sonyericsson.home.permission.BROADCAST_BADGE
- ✓ android.permission.CREDENTIAL_MANAGER_SET_ALLOWED_PROVIDERS
- ✓ android.permission.FOREGROUND_SERVICE_MICROPHONE
- ✓ com.facebook.pages.app.provider.ACCESS
- ✓ android.permission.USE_FULL_SCREEN_INTENT

- ✓ android.permission.FOREGROUND_SERVICE_CONNECTED_DEVICE
- ✓ com.android.vending.BILLING
- ✓ android.permission.NFC
- ✓ android.permission.MANAGE_ACCOUNTS
- ✓ android.permission.ACCESS_ADSERVICES_AD_ID
- ✓ android.permission.GET_TASKS
- ✓ android.permission.FOREGROUND_SERVICE_MEDIA_PROJECTION
- ✓ com.htc.launcher.permission.UPDATE_SHORTCUT
- ✓ android.permission.DOWNLOAD_WITHOUT_NOTIFICATION
- ✓ android.permission.RECEIVE_BOOT_COMPLETED
- ✓ android.permission.ACCESS_WIFI_STATE
- ✓ android.permission.WAKE_LOCK
- ✓ com.amazon.device.messaging.permission.RECEIVE
- ✓ com.facebook.mlite.provider.ACCESS
- ✓ com.sec.android.provider.badge.permission.READ
- ✓ android.permission.CHANGE_NETWORK_STATE
- ✓ android.permission.READ_SYNC_SETTINGS
- ✓ com.nokia.pushnotifications.permission.RECEIVE
- ✓ android.permission.READ_BASIC_PHONE_STATE
- ✓ android.permission.ACCESS_ADSERVICES_TOPICS
- ✓ android.permission.USE_FINGERPRINT
- ✓ android.permission.REORDER_TASKS
- ✓ com.google.android.c2dm.permission.RECEIVE
- ✓ android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
- ✓ android.permission.AUTHENTICATE_ACCOUNTS
- ✓ android.permission.FOREGROUND_SERVICE
- ✓ android.permission.VIBRATE
- ✓ android.permission.RUN_USER_INITIATED_JOBS
- ✓ android.permission.ACCESS_ADSERVICES_CUSTOM_AUDIENCE

Activities

The following activities were identified in the application:

- ✓ androidx.credentials.playservices.HiddenActivity
- ✓ com.android.billingclient.api.ProxyBillingActivity
- ✓ com.android.billingclient.api.ProxyBillingActivityV2
- ✓ com.facebook.about.AboutActivity
- ✓ com.facebook.abtest.gkprefs.GkSettingsListActivity
- ✓ com.facebook.abtest.qe.settings.QuickExperimentListActivity
- ✓ com.facebook.abtest.qe.settings.QuickExperimentViewActivity
- ✓ com.facebook.accessibility.central.debug.AccessibilityPlaygroundActivity

- ✓ com.facebook.account.bypasslogin.interstitial.BPLInterstitialActivity
- ✓ com.facebook.account.login.activity.SimpleLoginActivity
- ✓ com.facebook.account.recovery.AccountRecoveryActivity
- ✓ com.facebook.account.recovery.ear.ExtendedAccountRecoveryActivity
- ✓ com.facebook.account.securitykeys.playground.PlaygroundActivity
- ✓ com.facebook.account.sensitiveactionprotection.BloksSensitiveActionProtectionHostActivity
- ✓ com.facebook.account.simplerecovery.SimpleRecoveryActivity
- ✓ com.facebook.account.switcher.activity.SimpleAccountSwitcherActivity
- ✓ com.facebook.account.switcher.nux.ActivateDeviceBasedLoginNuxActivity
- ✓ com.facebook.account.switcher.settings.DBLLoggedInAccountSettingsActivity
- ✓ com.facebook.account.switcher.settings.DBLPinSettingsActivity
- ✓ com.facebook.account.switcher.shortcuts.AccountSwitcherShortcutActivity
- ✓ com.facebook.account.twofac.push.LoginApprovalsPushActivity
- ✓ com.facebook.accountquality.AccountQualityActivity
- ✓ com.facebook.accountquality.AccountQualityActorLauncherActivity
- ✓ com.facebook.accountquality.AccountQualityCommerceContentLauncherActivity
- ✓ com.facebook.actionitems.ActionItemsComposerActivity
- ✓ com.facebook.actorgateway.ui.ActorGatewayActivity
- ✓ com.facebook.addresstypeahead.AddressTypeAheadActivity
- ✓ com.facebook.adexpress.AdExpressActivity
- ✓ com.facebook.adinterfaces.AdInterfacesCouponInterstitialActivity
- ✓ com.facebook.adinterfaces.AdInterfacesExtendedObjectivesOfflineActivity
- ✓ com.facebook.adinterfaces.AdInterfacesObjectiveActivity
- ✓ com.facebook.adinterfaces.MapAreaPickerActivity
- ✓ com.facebook.adinterfaces.adcenter.AdCenterHostingActivity
- ✓ com.facebook.adinterfaces.adcenter.AdCenterPostSelectorHostingActivity
- ✓ com.facebook.adinterfaces.ui.selector.TargetingSelectorActivity
- ✓ com.facebook.adpreview.activity.AdPreviewActivity
- ✓ com.facebook.ads.AudienceNetworkActivity
- ✓ com.facebook.ads.internal.ipc.AudienceNetworkExportedActivity
- ✓ com.facebook.ads.internal.ipc.AudienceNetworkRemoteActivity
- ✓ com.facebook.adscomposer.AdsComposerActivity
- ✓ com.facebook.adsconsentvalue.acvdeeplinkhandler.ACVDeeplinkHandler
- ✓ com.facebook.adsexperienceTechnique.AdsInjectConfirmationActivity
- ✓ com.facebook.adspayments.activity.AddPaymentCardActivity
- ✓ com.facebook.adspayments.activity.AddressActivity
- ✓ com.facebook.adspayments.activity.BrazilianAddressActivity
- ✓ com.facebook.adspayments.activity.BrazilianTaxIdActivity
- ✓ com.facebook.adspayments.activity.IntentRouteHandlerActivity
- ✓ com.facebook.adspayments.activity.PaymentStatusActivity

- ✓ com.facebook.adspayments.activity.PrepayFlowFundingActivity
- ✓ com.facebook.adspayments.activity.SelectPaymentOptionActivity
- ✓ com.facebook.adspreviewinjection.AdsPreviewInjectionActivity
- ✓ com.facebook.analytics.fbmetricsdebugger.prefs.FbMetricsDebuggerSettingsActivity
- ✓ com.facebook.audience.snacks.storysurface.activity.StoriesSurfaceActivity
- ✓ com.facebook.audience.stories.archive.StoriesArchiveActivity
- ✓ com.facebook.audience.stories.highlights.StoriesHighlightsActivity
- ✓ com.facebook.audience.stories.highlights.editstoryhighlights.EditStoryHighlightsActivity
- ✓ com.facebook.audience.stories.highlights.highlightscreen.HighlightScreenActivity
- ✓ com.facebook.audience.stories.highlights.settings.StoriesHighlightsSettingsActivity
- ✓ com.facebook.auth.reauth.ReauthActivity
- ✓ com.facebook.authenticity.idv.IdentityVerificationActivity
- ✓ com.facebook.aymt.internalsettings.LookUpTipsActivity
- ✓ com.facebook.bladerunner_examples.FBBladeRunnerExamplesChatRoomPageActivity
- ✓ com.facebook.bladerunner_examples.FBBladeRunnerExamplesPageActivity
- ✓ com.facebook.bladerunner_examples.FBBladeRunnerHelloWorldPageActivity
- ✓ com.facebook.bladerunner_examples.FBBladeRunnerRealtimeClientSyncExamplePageActivity
- ✓ com.facebook.bladerunner_examples.FBBladeRunnerWeatherPageActivity
- ✓ com.facebook.blockchain_wallet.deeplinking.BlockchainWalletUriHandlerActivity
- ✓ com.facebook.bloks.facebook.actions.plugins.bkactionfbfbcontextualexperiencecreateandissuenativewalletoffer.CreateAndIssueNativeWalletOfferActivity
- ✓ com.facebook.bloks.facebook.actions.plugins.bkactionlaunchsecureembeddedwebview.WebAuthenticationActivity
- ✓ com.facebook.bloks.facebook.actions.plugins.bkactionsecurewebviewwithonchange.TrustlyAuthenticationActivity
- ✓ com.facebook.bloks.facebook.actions.plugins.bkactiontrustlyauthentication.FBRedirectUrlHandlerActivity
- ✓ com.facebook.bloks.facebook.actions.plugins.bkactiontrustlyauthentication.TrustlyLightboxActivity
- ✓ com.facebook.bloks.facebook.actions.plugins.bkactionwebviewwithonchange.OpenWebViewWithUrlChangeActivity
- ✓ com.facebook.bloks.facebook.loggedout.FbExperimentalLoggedOutBloksActivity
- ✓ com.facebook.bloks.facebook.nativeshell.FbBloksNativeShellActivity
- ✓ com.facebook.bloks.messenger.activity.MSGBloksActivity

- ✓ com.facebook.bloks.messenger.activity.MSGBloksTransparentActivity
- ✓ com.facebook.brandfunded.branddeals.deeplink.FBCreatorMarketplacePermalinkUriHandlerActivity
- ✓ com.facebook.brandfunded.deeplink.AffiliatePartnershipsPermalinkUriHandlerActivity
- ✓ com.facebook.browser.lite.BrowserLite2Activity
- ✓ com.facebook.browser.lite.BrowserLiteActivity
- ✓ com.facebook.browser.lite.BrowserLiteInMainProcessActivity
- ✓ com.facebook.browser.lite.BrowserLiteInMainProcessBottomSheetActivity
- ✓ com.facebook.browser.lite.RecentTabsActivity
- ✓ com.facebook.browser.lite.debug.debugger.BrowserLiteDebuggerActivity
- ✓ com.facebook.browser.lite.debug.debugger.BrowserLiteDebuggerSettingsActivity
- ✓ com.facebook.browser.lite.extensions.autofill.facebook.internalsettings.AutofillKeyMigrationPlaygroundActivity
- ✓ com.facebook.browser.liteclient.rapidfeedback.BrowserLiteRapidFeedbackActivity
- ✓ com.facebook.browser.liteclient.report.BrowserRapidReportingHostActivity
- ✓ com.facebook.browser.liteclient.report.BrowserReportingInterstitialActivity
- ✓ com.facebook.browser.litev2.lite.BrowserLiteDIActivity
- ✓ com.facebook.browser.litev2.lite.BrowserLiteDITransparentActivity
- ✓ com.facebook.bugreporter.activity.BugReportActivity
- ✓ com.facebook.bugreporter.activity.suggestifier.nonwork.SuggestifierActivity
- ✓ com.facebook.bugreporter.mediacapture.BugReportMediaCaptureActivity
- ✓ com.facebook.bugreporter.screenshot.uipresenter.BugReportScreenshotActivity
- ✓ com.facebook.businessintegrity.gdpr.consents.GDPRConsentsActivity
- ✓ com.facebook.businessintegrity.mlex.adactivity.dashboard.AdActivityDashboardActivity
- ✓ com.facebook.businessintegrity.waist.BIWaistActivity
- ✓ com.facebook.businesslinkmanagement.BusinessLinkManagementSettingEntrypointActivity

Services

The following services were identified in the application:

- ✓ androidx.credentials.playservices.CredentialProviderMetadataHolder
- ✓ androidx.room.MultiInstanceInvalidationService
- ✓ androidx.work.impl.background.systemalarm.SystemAlarmService
- ✓ androidx.work.impl.background.systemjob.SystemJobService
- ✓ androidx.work.impl.foreground.SystemForegroundService
- ✓ androidx.work.multiprocess.RemoteWorkManagerService

- ✓ com.facebook.account.login.notification.LoginNotificationService
- ✓ com.facebook.account.twofac.protocol.LoginApprovalNotificationService
- ✓ com.facebook.adinterfaces.service.BoostLiveService
- ✓ com.facebook.ads.internal.ipc.AudienceNetworkRemoteService
- ✓ com.facebook.adspayments.analytics.ExperimentExposeService
- ✓ com.facebook.analytics2.logger.GooglePlayUploadService
- ✓ com.facebook.analytics2.logger.legacy.uploader.AlarmBasedUploadService
- ✓ com.facebook.analytics2.logger.legacy.uploader.Analytics2UploadService
- ✓ com.facebook.analytics2.logger.legacy.uploader.LollipopUploadService
- ✓ com.facebook.analytics2.logger.service.LollipopUploadSafeService
- ✓ com.facebook.appcomponentmanager.AppComponentManagerService
- ✓ com.facebook.audiencenetwork.AudienceNetworkService
- ✓ com.facebook.battery.monitor.ContinuousBatteryMonitorService
- ✓ com.facebook.browser.helium.content.SandboxedProcessService0
- ✓ com.facebook.browser.helium.content.SandboxedProcessService1
- ✓ com.facebook.browser.helium.preload.AppZygoteWarmerService
- ✓ com.facebook.browser.lite.BrowserLiteInMainProcessIntentService
- ✓ com.facebook.browser.lite.BrowserLiteIntentService
- ✓ com.facebook.browser.liteclient.BrowserLiteCallbackService
- ✓ com.facebook.bugreporter.core.scheduler.GCMBugReportService
- ✓ com.facebook.bugreporter.core.scheduler.LollipopBugReportService
- ✓ com.facebook.businessintegrity.cloakingdetection.service.CloakingDetectionService
- ✓ com.facebook.cameracore.mediapipeline.asyncscripting.AsyncScriptingService
- ✓ com.facebook.cid.server.LocalProcessOnDeviceSignalServiceHost
- ✓ com.facebook.cid.server.RemoteProcessOnDeviceSignalServiceHost
- ✓ com.facebook.common.errorreporting.memory.service.jobschedulercompat.fbsvc.DumperUploadService
- ✓ com.facebook.common.zopt.ZOptBackgroundService
- ✓ com.facebook.composer.publish.internal.ComposerPublishService
- ✓ com.facebook.compost.service.CompostNotificationService
- ✓ com.facebook.conditionalworker.ConditionalWorkerService
- ✓ com.facebook.conditionalworker.GooglePlayConditionalWorkerService
- ✓ com.facebook.conditionalworker.LollipopConditionalWorkerService
- ✓ com.facebook.confirmation.notification.ConfirmationNotificationService
- ✓ com.facebook.confirmation.service.AddPhoneNumberService
- ✓ com.facebook.confirmation.service.AutoSmsConfirmService
- ✓ com.facebook.contacts.service.ContactLocaleChangeService
- ✓ com.facebook.delayedworker.DelayedWorkerService
- ✓ com.facebook.directinstall.feed.progressservice.InstallNotificationService
- ✓ com.facebook.directinstall.feed.progressservice.ProgressService
- ✓ com.facebook.directinstall.shortcut.ShortcutInstallerService

- ✓ com.facebook.facecast.broadcast.notifications.LiveAudioRoomV2NotificationService
- ✓ com.facebook.facecast.display.debugoverlay.FacecastDebugOverlayService
- ✓ com.facebook.fbpay.w3c.ipc.FBPaymentServiceImpl
- ✓ com.facebook.fbpay.w3c.ipc.IsReadyToPayServiceImpl
- ✓ com.facebook.fbservice.service.BlueService
- ✓ com.facebook.fbservice.service.BlueServiceJobIntentService
- ✓ com.facebook.feed.platformads.AppInstallService
- ✓ com.facebook.feed.platformads.AppInstallTrackerService
- ✓ com.facebook.fixie.fixes.signal.AppDeathSignalService
- ✓ com.facebook.friending.jewel.FriendRequestsPrivacySettingService
- ✓ com.facebook.glc.GLCService
- ✓ com.facebook.glc.LongRunningWifiMonitorService
- ✓ com.facebook.glc.ScheduledWifiMonitorService
- ✓ com.facebook.googleplay.GooglePlayInstallRefererService
- ✓ com.facebook.graphql.executor.offlinemutations.OfflineMutationsRetryGCMTaskService
- ✓ com.facebook.graphql.executor.offlinemutations.OfflineMutationsRetryJobService
- ✓ com.facebook.huddle.notification.speakernotification.impl.HuddleLiveSessionNotificationService
- ✓ com.facebook.katana.RemoveRawContactsService
- ✓ com.facebook.katana.platform.FacebookAuthenticationService
- ✓ com.facebook.katana.platform.TokenRefreshService
- ✓ com.facebook.lite.webviewrtc.RTCService
- ✓ com.facebook.media.local.UpdateLocalMediaStoreGcmTaskService
- ✓ com.facebook.media.local.UpdateLocalMediaStoreJobService
- ✓ com.facebook.messaging.notify.logging.MessagesSystemTrayLogService
- ✓ com.facebook.messaging.prefs.notifications.NotificationPrefsSyncService
- ✓ com.facebook.mqtt.service.MqttServiceV2
- ✓ com.facebook.mqttlite.MqttService
- ✓ com.facebook.multiusermqtt.MultiuserMqttService
- ✓ com.facebook.notifications.appwidget.bugreporter.NotificationsWidgetDebugHelper
- ✓ com.facebook.notifications.appwidget.refresh.NotificationsWidgetCollectionService
- ✓ com.facebook.notifications.tray.actions.PushNotificationsActionService
- ✓ com.facebook.notifications.tray.service.SystemTrayLogService
- ✓ com.facebook.orca.notify.MessagesNotificationService
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppCacheJobService

- ✓ com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppCacheService
- ✓ com.facebook.papaya.fb.client.services.FBPapayaJobService
- ✓ com.facebook.photos.upload.service.PhotosUploadHelperService
- ✓ com.facebook.platform.common.service.PlatformService
- ✓ com.facebook.ppml.receiver.ReceiverService
- ✓ com.facebook.profilo.upload.TraceUploadRetryJob
- ✓ com.facebook.pumpkin.PumpkinService
- ✓ com.facebook.push.adm.ADMBroadcastReceiver
- ✓ com.facebook.push.adm.ADMBroadcastReceiverJobBase
- ✓ com.facebook.push.adm.ADMJobIntentService
- ✓ com.facebook.push.adm.ADMService
- ✓ com.facebook.push.adm.AdmWorkGCMService
- ✓ com.facebook.push.adm.AdmWorkLollipopService
- ✓ com.facebook.push.fbpushdata.common.FbPushDataHandlerService
- ✓ com.facebook.push.fcm.FcmListenerService
- ✓ com.facebook.push.fcm.FcmRegistrarFbJobIntentService
- ✓ com.facebook.push.fcm.GetFcmTokenRegistrarGCMService
- ✓ com.facebook.push.fcm.GetFcmTokenRegistrarLollipopService
- ✓ com.facebook.push.mqtt.service.MqttPushHelperService
- ✓ com.facebook.push.negativefeedback.PushNegativeFeedbackFbJobIntentService

Receivers

The following receivers were identified in the application:

- ✓ androidx.profileinstaller.ProfileInstallReceiver
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$StorageNotLowProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxyUpdateReceiver
- ✓ androidx.work.impl.background.systemalarm.RescheduleReceiver
- ✓ androidx.work.impl.diagnostics.DiagnosticsReceiver
- ✓ androidx.work.impl.utils.ForceStopRunnable$BroadcastReceiver
- ✓ com.facebook.account.login.notification.LoginNotificationServiceReceiver
- ✓ com.facebook.account.login.notification.LoginNotificationServiceScheduler

- ✓ com.facebook.account.switcher.shortcuts.AccountSwitcherShortcutReceiver
- ✓ com.facebook.analytics2.logger.legacy.uploader.HighPriUploadRetryReceiver
- ✓ com.facebook.appcomponentmanager.fb.FbAppComponentReceiver
- ✓ com.facebook.appcomponentmanager.testreceivers.AppComponentManagerTestingReceiver
- ✓ com.facebook.appcomponentmanager.testreceivers.FirstEnableStageTestReceiver
- ✓ com.facebook.appcomponentmanager.testreceivers.SecondEnableStageTestReceiver
- ✓ com.facebook.battery.monitor.ContinuousBatteryMonitorService$BroadcastReceiver
- ✓ com.facebook.bugreporter.core.scheduler.AlarmsBroadcastReceiver
- ✓ com.facebook.caa.shared.identityhash.IdentityHashReceiver
- ✓ com.facebook.caa.shared.wacoderetriever.CAARegWACodeReceiver
- ✓ com.facebook.clicktocall.CTCPhoneCallBroadcastReceiver
- ✓ com.facebook.clicktocall.LeadAdsP2BCallStateChangeReceiver
- ✓ com.facebook.common.init.OnAppUpdateReceiver
- ✓ com.facebook.compost.service.CompostNotificationServiceReceiver
- ✓ com.facebook.conditionalworker.ConditionalWorkerServiceReceiver
- ✓ com.facebook.confirmation.notification.ConfirmationNotificationReScheduler
- ✓ com.facebook.confirmation.notification.ConfirmationNotificationServiceReceiver
- ✓ com.facebook.confirmation.task.SMSBroadcastReceiver
- ✓ com.facebook.confirmation.util.VoiceCallListenerExpiryReceiver
- ✓ com.facebook.contacts.service.ContactLocaleChangeReceiver
- ✓ com.facebook.credentials.AuthTokenDebugLogger
- ✓ com.facebook.delayedworker.DelayedWorkerServiceReceiver
- ✓ com.facebook.device_id.UniqueIdSupplier
- ✓ com.facebook.errorreporting.lacrima.detector.broadcast.ProtectedLockScreenBroadcastReceiver
- ✓ com.facebook.errorreporting.lacrima.detector.broadcast.PublicLockScreenBroadcastReceiver
- ✓ com.facebook.errorreporting.lacrima.detector.broadcast.SecureShutdownBootBroadcastReceiver
- ✓ com.facebook.fbreact.neo.ReactParentDashboardShortcutModule$ParentDashboardShortcutInstalledIntentReceiver
- ✓ com.facebook.fdidlite.FDIDSyncLiteReceiver
- ✓ com.facebook.feed.platformads.AppInstallReceiver
- ✓ com.facebook.feedplugins.share.externalshare.logging.FeedShareIntentResultBroadcastReceiver
- ✓ com.facebook.games.entrypoint.shortcut.GamesShortcutHelper$GamingTabBookmarkShortcutCreatedIntentReceiver

- ✓ com.facebook.glc.GLCServiceSchedulerReceiver
- ✓ com.facebook.googleplay.GooglePlayInstallReferrerReceiver
- ✓ com.facebook.groups.share.helpers.GroupOpenChooserShareIntentBroadcastReceiver
- ✓ com.facebook.katana.FacebookAccountReceiver
- ✓ com.facebook.messaging.messengerprefs.notif.NoteNotificationBroadcastReceiver
- ✓ com.facebook.messaging.notify.util.intent.MessengerNotificationAndroidSurveyReceiver
- ✓ com.facebook.messaging.notify.util.intent.NoteNotificationMuteBroadcastReceiver
- ✓ com.facebook.messaginginblue.threadview.ui.activity.main.BubblesDeleteIntentReceiver
- ✓ com.facebook.mobileconfig.init.MobileConfigEnableReceiver
- ✓ com.facebook.nobreak.CrashLoop$LastState
- ✓ com.facebook.notifications.appwidget.MediumNotificationsWidgetProvider
- ✓ com.facebook.notifications.appwidget.SmallNotificationsWidgetProvider
- ✓ com.facebook.notifications.appwidget.receiver.NotificationsWidgetAppUpgradeReceiver
- ✓ com.facebook.notifications.appwidget.widgetpicker.NotificationsWidgetPickerPinnedReceiver
- ✓ com.facebook.notifications.push.token.refresh.NotificationsTokenRefreshReceiver
- ✓ com.facebook.notifications.tray.helpers.receivers.ActivityTransitionReceiver
- ✓ com.facebook.notifications.tray.helpers.receivers.NotificationsRemovalAppUpdateReceiver
- ✓ com.facebook.notifications.tray.helpers.receivers.NotificationsRemovalTimerReceiver
- ✓ com.facebook.notifications.tray.util.NotificationsClientSignalStaticBroadcastReceiver
- ✓ com.facebook.orca.notify.MessagesNotificationBroadcastReceiver
- ✓ com.facebook.oxygen.preloads.integration.dogfooding.AuthListener$EmployeeLoggedInMarkerReceiver
- ✓ com.facebook.oxygen.preloads.integration.sso.PostInstallSsoReceiver
- ✓ com.facebook.oxygen.preloads.integration.tosacceptance.TosAcceptedFlagComponent
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppFlag
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppReceiver
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.settings.TosAcceptedFlag
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.settings.wa.ITA

- ✓ com.facebook.photos.upload.receiver.ConnectivityChangeReceiver
- ✓ com.facebook.push.adm.MessageAlertReceiver
- ✓ com.facebook.push.fbnslite.FbPushSdkFbnsReceiverShim
- ✓ com.facebook.push.fbnslite.FbnsLiteRtcPushNotificationReceiver
- ✓ com.facebook.push.mqtt.receiver.BootCompleteBroadcastReceiver
- ✓ com.facebook.push.negativefeedback.PushNegativeFeedbackReceiver
- ✓ com.facebook.push.nna.NNABroadcastReceiver
- ✓ com.facebook.push.registration.FacebookPushServerRegistrar$LocalBroadcastReceiver
- ✓ com.facebook.push.registration.RegistrarHelperReceiver
- ✓ com.facebook.quicksilver.util.QuicksilverShortcutCreationUtil$ShortcutMadeIntentReceiver
- ✓ com.facebook.registration.notification.RegistrationNotificationServiceReceiver
- ✓ com.facebook.rooms.call.receivers.PictureInPictureReceiver
- ✓ com.facebook.rooms.call.receivers.RoomEndCallReceiver
- ✓ com.facebook.rooms.mainapp.receivers.RoomsStartCallReceiver
- ✓ com.facebook.rtc.notification.NotificationDeletedBroadcastReceiver
- ✓ com.facebook.rtc.notification.RtcMultiCallActionReceiver
- ✓ com.facebook.rtc.notification.metaai.MetaAiVoiceBroadcastReceiver
- ✓ com.facebook.rti.orca.FbnsLiteBroadcastReceiver
- ✓ com.facebook.rti.orca.UpdateQeBroadcastReceiver
- ✓ com.facebook.ui.mainview.tabshortcut.TabShortcutCreator$ShortcutMadeIntentReceiver
- ✓ com.facebook.uievaluations.uiqr.fbcore.sapienz.TriggerUIEvaluationsIntentReceiver
- ✓ com.facebook.watson.WatsonBroadcastReceiver
- ✓ com.facebook.whatsapp.foabackuptoken.FoaBackupTokenReceiver
- ✓ com.facebook.zero.easydogfooding.notification.ZeroEasyDogfoodNotificationActionReceiver
- ✓ com.facebook.zero.iptest.ZeroIPTestBroadcastReceiver
- ✓ com.google.android.gms.analytics.AnalyticsReceiver
- ✓ com.google.firebase.iid.FirebaseInstanceIdReceiver

Providers

The following providers were identified for the application:

- ✓ androidx.core.content.FileProvider
- ✓ androidx.startup.InitializationProvider
- ✓ com.facebook.abtest.qe.db.QuickExperimentContentProvider
- ✓ com.facebook.contacts.data.FbContactsContentProvider
- ✓ com.facebook.contacts.provider.ContactsConnectionsProvider

- ✓ com.facebook.fdidlite.FDIDLiteProvider
- ✓ com.facebook.katana.liteprovider.FirstPartyUserValuesLiteProvider
- ✓ com.facebook.katana.liteprovider.usdid.UsdidValuesProvider
- ✓ com.facebook.katana.provider.AttributionIdProvider
- ✓ com.facebook.katana.provider.CacheProvider
- ✓ com.facebook.katana.provider.FirstPartyUserValuesProvider
- ✓ com.facebook.katana.provider.InstallReferrerProvider
- ✓ com.facebook.katana.provider.UserValuesProvider
- ✓ com.facebook.messaging.database.threads.MessagesDbContentProvider
- ✓ com.facebook.messaging.database.threads.ThreadsDbPropertiesContentProvider
- ✓ com.facebook.messaging.push.dedup.provider.ClientMessagePushDedupInfoProvider
- ✓ com.facebook.messaging.sharekey.contentprovider.SecureMessagingKeyContentProvider
- ✓ com.facebook.mobileidservices.feo2.fb4a.fbprovider.FeO2InAppContentProvider
- ✓ com.facebook.oxygen.preloads.integration.dogfooding.AppManagerSsoProvider
- ✓ com.facebook.photos.provider.PhotosProvider
- ✓ com.facebook.platform.common.provider.PlatformProviderBase
- ✓ com.facebook.push.fcm.customprovider.FirebaseInitCustomProvider
- ✓ com.facebook.rti.common.sharedprefs.SharedPreferencesContentProvider
- ✓ com.facebook.secure.fileprovider.SecureFileProvider
- ✓ com.facebook.whatsapp.foabackuptoken.FoaBackupTokenProvider
- ✓ com.facebook.xapp.messaging.composer.broadcast.transportagnostic.attachment.source.TamAttachmentProvider

Intent Filter by Action

- ✓ android.accounts.AccountAuthenticator
- ✓ com.meta.trusteddevice.TRUSTED_DEVICE_FOUNDATION_SERVICE
- ✓ com.fbpay.w3c.GET_DEVICE_KEY
- ✓ com.google.android.gms.gcm.ACTION_TASK_READY
- ✓ com.facebook.platform.PLATFORM_SERVICE
- ✓ com.google.android.gms.wearable.BIND_LISTENER
- ✓ android.telecom.ConnectionService
- ✓ com.facebook.huddle.notification.speakernotification.impl.HuddleLiveSessionNotificationService
- ✓ com.google.firebase.MESSAGING_EVENT
- ✓ com.facebook.video.tv.mediasession.TVCastingMediaSessionService
- ✓ com.facebook.video.tv.notification.TV_DISCONNECT_ACTION

- ✓ com.facebook.video.tv.notification.TV_PLAY_PAUSE_ACTION
- ✓ com.facebook.video.tv.notification.TV_CLICK_ACTION
- ✓ com.facebook.adinterfaces.service.BOOST_LIVE
- ✓ ReceiverService
- ✓ com.facebook.video.watchandgo.service.UnifiedMiniPlayerService
- ✓ org.chromium.intent.action.IS_READY_TO_PAY
- ✓ com.facebook.facecast.broadcast.notifications.LiveAudioRoomV2NotificationService
- ✓ com.facebook.video.bgplayback.notification.service.PlaybackNotificationService
- ✓ com.fbpay.w3c.FB_EXTENSIONS
- ✓ com.meta.wearable.acdc.sdk.service.ACDCRegistrationService.BIND
- ✓ com.facebook.audiencenetwork.AUDIENCE_NETWORK_VERIFY
- ✓ com.facebook.video.watchandgo.service.WatchAndGoService
- ✓ com.facebook.browser.lite.BrowserLiteCallback
- ✓ com.facebook.games.gaming_services.DEEPLINK
- ✓ android.intent.action.VIEW
- ✓ com.facebook.aymt.internalsettings.LOOK_UP_TIPS_ACTIVITY
- ✓ com.fbpay.w3c.PAYMENT_METHODS_SETTINGS
- ✓ org.chromium.intent.action.PAY
- ✓ com.facebook.intent.action.prod.VIEW_TRANSPARENT_PERMALINK
- ✓ com.facebook.orca.STAX_THREAD_VIEW_BUBBLE
- ✓ com.facebook.intent.action.prod.VIEW_PERMALINK
- ✓ com.facebook.intent.action.prod.VIEW_COLLECTION
- ✓ com.facebook.intent.action.prod.VIEW_NOTIFICATION_SETTINGS
- ✓ com.google.android.gms.actions.SEARCH_ACTION
- ✓ com.facebook.share.ADD_FROM_BASEL
- ✓ android.intent.action.BATTERY_OKAY
- ✓ android.intent.action.BATTERY_LOW
- ✓ android.intent.action.BOOT_COMPLETED
- ✓ android.intent.action.MY_PACKAGE_REPLACED
- ✓ android.intent.action.ACTION_POWER_DISCONNECTED
- ✓ android.intent.action.ACTION_POWER_CONNECTED
- ✓ android.intent.action.AIRPLANE_MODE
- ✓ android.intent.action.SCREEN_ON
- ✓ android.intent.action.SCREEN_OFF
- ✓ com.facebook.credentials.AuthTokenDebugLogger.Trigger
- ✓ android.accounts.LOGIN_ACCOUNTS_CHANGED
- ✓ android.appwidget.action.APPWIDGET_UPDATE
- ✓ android.appwidget.action.APPWIDGET_ENABLED
- ✓ android.appwidget.action.APPWIDGET_DISABLED
- ✓ android.appwidget.action.APPWIDGET_UPDATE_OPTIONS

- ✓ android.appwidget.action.APPWIDGET_RESTORED
- ✓ android.appwidget.action.APPWIDGET_ENABLE_AND_UPDATE
- ✓ com.facebook.android.appwidget.action.APPWIDGET_UPDATE
- ✓ com.facebook.android.appwidget.action.APPWIDGET_ENABLED
- ✓ com.facebook.android.appwidget.action.APPWIDGET_DISABLED
- ✓ com.facebook.android.appwidget.action.APPWIDGET_UPDATE_OPTIONS
- ✓ com.facebook.android.appwidget.action.APPWIDGET_RESTORED
- ✓ com.facebook.android.appwidget.action.APPWIDGET_ENABLE_AND_UPDATE
- ✓ com.facebook.notifications.appwidget.ACTION_FORCE_UPDATE
- ✓ com.facebook.intent.action.prod.FOR_LOGIN_NOTIFICATION_SERVICE
- ✓ com.facebook.uievaluations.uiqr.fbcore.sapienz.SAPIENZ_RUN_UI_EVALUATION
- ✓ android.intent.action.PHONE_STATE
- ✓ com.facebook.push.registration.ACTION_TOKEN_REQUEST_RETRY
- ✓ com.facebook.push.registration.ACTION_TOKEN_REQUEST_RETRY_GCM
- ✓ com.facebook.push.registration.ACTION_TOKEN_REQUEST_RETRY_FBNS
- ✓ com.facebook.intent.action.prod.ROOMS_ACCEPT_CALL_ACTION
- ✓ com.facebook.intent.action.prod.ROOMS_DECLINE_CALL_ACTION
- ✓ com.facebook.intent.action.prod.ROOMS_SHOW_CURRENT_CALL_ACTION
- ✓ com.facebook.intent.action.prod.ROOM_MODULE_DOWNLOAD_COMPLETE
- ✓ com.facebook.intent.action.prod.FOR_CONDITIONAL_WORKER_SERVICE
- ✓ com.facebook.notifications.appwidget.ACTION_CONTENT_TAPPED
- ✓ com.facebook.notifications.appwidget.ACTION_ONEWIDGET_TAB_SELECTOR_TAPPED
- ✓ android.intent.action.TIME_SET
- ✓ android.intent.action.TIMEZONE_CHANGED
- ✓ android.net.conn.CONNECTIVITY_CHANGE
- ✓ com.facebook.intent.action.prod.ACTION_FB_BUBBLE_DELETED
- ✓ com.facebook.account.switcher.shortcuts.SHORTCUT_CREATED
- ✓ com.facebook.games.entrypoint.shortcut.CREATE_GAMING_TAB_BOOKMARK_SHORTCUT
- ✓ com.facebook.android.appwidget.action.WIDGET_PICKER_SUCCESS
- ✓ androidx.profileinstaller.action.INSTALL_PROFILE
- ✓ androidx.profileinstaller.action.SKIP_FILE
- ✓ androidx.profileinstaller.action.SAVE_PROFILE
- ✓ androidx.profileinstaller.action.BENCHMARK_OPERATION
- ✓ com.facebook.intent.action.prod.FOR_COMPOST_NOTIFICATION_SERVICE

- ✓ com.whatsapp.otp.OTP_RETRIEVED
- ✓ com.facebook.appcomponentmanager.ACTION_TEST
- ✓ com.facebook.appcomponentmanager.ACTION_PRINT_COMPONENTS
- ✓ androidx.work.diagnostics.REQUEST_DIAGNOSTICS
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IS_MANAGED_APP_CHANGED
- ✓ com.facebook.intent.action.prod.PICTURE_IN_PICTURE_MUTE_TOGGLE
- ✓ com.facebook.intent.action.prod.PICTURE_IN_PICTURE_VIDEO_TOGGLE
- ✓ com.facebook.intent.action.prod.PICTURE_IN_PICTURE_END_CALL
- ✓ com.facebook.appcomponentmanager.ENABLING_CMPS_DONE
- ✓ com.facebook.GET_UNIQUE_ID
- ✓ com.facebook.quicksilver.CREATE_SHORTCUT
- ✓ com.facebook.intent.action.prod.PUSH_NEGATIVE_FEEDBACK_SERVICE
- ✓ com.facebook.GET_PHONE_ID
- ✓ com.facebook.navigation.tabbar.appshortcut.CREATE_SHORTCUT
- ✓ androidx.work.impl.background.systemalarm.UpdateProxies
- ✓ com.facebook.intent.action.prod.ROOMS_END_CALL_ACTION
- ✓ com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppFlag
- ✓ com.google.android.gms.auth.api.phone.SMS_RETRIEVED
- ✓ com.instagram.android.intent.action.ACTION_SCREEN_ON
- ✓ com.instagram.android.intent.action.ACTION_SCREEN_OFF
- ✓ com.facebook.bugreporter.core.scheduler.AlarmsBroadcastReceiver.RETRY_UPLOAD
- ✓ com.facebook.android.intent.action.BOOT_COMPLETED
- ✓ com.facebook.android.intent.action.MY_PACKAGE_REPLACED
- ✓ com.facebook.intent.action.prod.FOR_REGISTRATION_NOTIFICATION_SERVICE
- ✓ com.nokia.pushnotifications.intent.RECEIVE
- ✓ com.nokia.pushnotifications.intent.REGISTRATION
- ✓ com.facebook.rtc.notification.DECLINE_MULTI_CALL_ACTION
- ✓ com.android.vending.INSTALL_REFERRER
- ✓ com.facebook.orca.notify.ACTION_NEW_MESSAGE
- ✓ com.facebook.orca.notify.ACTION_FAILED_SEND
- ✓ com.facebook.orca.notify.ACTION_NEW_FRIEND_INSTALL
- ✓ com.facebook.orca.notify.ACTION_P2P_PAYMENT
- ✓ com.facebook.orca.notify.ACTION_LOGGED_OUT_MESSAGE
- ✓ com.facebook.orca.notify.ACTION_CLEAR_THREAD
- ✓ com.facebook.orca.notify.ACTION_CLEAR_FRIEND_INSTALL
- ✓ com.facebook.orca.notify.ACTION_CLEAR_ALL_NOTIFICATIONS
- ✓ com.facebook.orca.notify.ACTION_INTERNAL_MESSAGE
- ✓ com.facebook.orca.notify.ACTION_PRE_REG_PUSH_MESSAGE

- ✓ com.facebook.intent.action.prod.FOR_DELAYED_WORKER_SERVICE
- ✓ com.facebook.push.registration.ACTION_ALARM
- ✓ com.facebook.fbreact.neo.INSTALL_PD_SHORTCUT_SUCCESS
- ✓ android.intent.action.ACTION_SHUTDOWN
- ✓ com.facebook.intent.action.prod.ZERO_IP_TEST_ACTION
- ✓ android.intent.action.USER_PRESENT
- ✓ com.facebook.rti.mqtt.intent.ACTION_WAKEUP
- ✓ android.intent.action.PACKAGE_ADDED
- ✓ com.facebook.rti.fbns.intent.RECEIVE_RTC
- ✓ android.intent.action.DEVICE_STORAGE_LOW
- ✓ android.intent.action.DEVICE_STORAGE_OK
- ✓ com.facebook.SET_FOA_BACKUP_TOKEN
- ✓ com.facebook.GET_FOA_BACKUP_TOKEN
- ✓ com.facebook.rti.fbns.intent.RECEIVE
- ✓ com.whatsapp.otp.ID_HASH_REQUESTED
- ✓ com.google.android.c2dm.intent.RECEIVE
- ✓ android.intent.action.ACTION_SCREEN_ON
- ✓ android.intent.action.ACTION_SCREEN_OFF
- ✓ com.facebook.appmanager.POST_INSTALL_SSO
- ✓ com.facebook.messaging.messengerprefs.notif.ACTION_MUTE_NOTE_NOTIFICATION
- ✓ com.amazon.device.messaging.intent.REGISTRATION
- ✓ com.amazon.device.messaging.intent.RECEIVE
- ✓ com.facebook.messaging.notify.util.intent.MESSENGER_NOTIFICATION_NOTE_MUTE
- ✓ com.facebook.rtc.notification.metaai.END_SESSION_ACTION
- ✓ com.facebook.rtc.notification.metaai.TOGGLE_MUTE_ACTION
- ✓ com.facebook.messaging.notify.util.intent.MESSENGER_NOTIFICATION_ANDROID_SURVEY_USEFUL_ACTION
- ✓ com.facebook.messaging.notify.util.intent.MESSENGER_NOTIFICATION_ANDROID_SURVEY_NOT_USEFUL_ACTION
- ✓ android.intent.action.LOCALE_CHANGED
- ✓ com.instagram.android.intent.action.ACTION_SHUTDOWN
- ✓ com.instagram.android.intent.action.BOOT_COMPLETED
- ✓ com.facebook.intent.action.prod.FOR_CONFIRMATION_NOTIFICATION_SERVICE

Intent Filter by Category

- ✓ android.intent.category.DEFAULT
  - ❖ com.facebook.platform.common.service.PlatformService
  - ❖ com.facebook.ppml.receiver.ReceiverService

- ❖ com.meta.wearable.acdc.sdk.service.ACDCRegistrationService
- ❖ com.facebook.games.gamingservices.deeplink.GamingServicesDeeplinkActivity
- ❖ com.facebook.languages.switcher.nonworkactivity.LanguageSwitchPromotionActivity
- ❖ com.facebook.fbpay.w3c.views.PaymentMethodsActivity
- ❖ com.facebook.katana.UriAuthHandler
- ❖ com.facebook.katana.immersiveactivity.ImmersiveTransparentActivity
- ❖ com.facebook.adinterfaces.AdInterfacesObjectiveActivity
- ❖ com.facebook.katana.IntentUriHandler

✓ android.intent.category.BROWSABLE

- ❖ com.facebook.katana.UriAuthHandler
- ❖ com.facebook.adinterfaces.AdInterfacesObjectiveActivity
- ❖ com.facebook.katana.IntentUriHandler
- ❖ com.facebook.fx.thirdpartyoauth.FxFbThirdPartyOAuthActivity

✓ com.facebook.katana

- ❖ com.facebook.push.nna.NNABroadcastReceiver
- ❖ com.facebook.push.fbnslite.FbnsLiteRtcPushNotificationReceiver
- ❖ com.facebook.push.fbnslite.FbPushSdkFbnsReceiverShim
- ❖ com.facebook.push.adm.MessageAlertReceiver

Contacted URLs

Only one URL was being used in the contact process:

Item	Description
Scanned Date	6/7/2025
Detections	0
Total Scanners	97
Status	204
URL	http://connectivitycheck.gstatic.com/generate_204

Contacted Domains

The following domains were being contacted by the Facebook application:

Domain	Detections	Created	Registrar
b-www.facebook.com	0 / 94	3/29/1997	RegistrarSafe, LLC
connectivitycheck.gstatic.com	0 / 94	2/11/2008	MarkMonitor Inc.
gstatic.com	0 / 94	2/11/2008	MarkMonitor Inc.
star-mini.c10r.facebook.com	0 / 94	3/29/1997	RegistrarSafe, LLC

Contacted IP Addresses

The following IP addresses were being contacted by the application. They were found to be secure in all the 94 analyzed repositories:

IP	Detections	Autonomous System	Country
104.21.64.137	0 / 94	13335	-
108.177.121.104	0 / 94	15169	US
142.251.163.94	0 / 94	15169	US
142.251.163.95	0 / 94	15169	US
157.240.241.35	0 / 94	32934	US
172.253.122.105	0 / 94	15169	US
172.67.181.52	0 / 94	13335	-
173.194.193.138	0 / 94	15169	US
173.194.193.139	0 / 94	15169	US
173.194.193.95	0 / 94	15169	US
173.194.194.94	0 / 94	15169	US
173.194.206.106	0 / 94	15169	US
173.194.206.95	0 / 94	15169	US
209.85.145.95	0 / 94	15169	US
64.233.179.95	0 / 94	15169	US
74.125.202.104	0 / 94	15169	US

Bundled files

The following bundled files were obtained in the application:

Scanned	Detections	File type	Name
5/30/2025	0 / 61	Text	LICENSE_OFL

4/23/2025	0 / 61	Text	LICENSE_UNICODE
7/10/2015	0 / 56	Text	META-INF/androidx.compose.ui_ui.version
5/31/2025	0 / 62	Text	META-INF/services/kotlinx.coroutines.CoroutineExceptionHandler
8/25/2020	0 / 57	Text	NOTICE
5/27/2025	0 / 62	Text	androidsupportmultidexversion.txt
4/23/2025	0 / 61	INI	app-update.properties
5/29/2025	0 / 60	Text	assets/added_params_map.txt
1/11/2022	0 / 57	Text	assets/api_key.txt
6/5/2025	0 / 62	JSON	assets/app_modules/contents/eventsguestlist.json

IP Traffic

IP traffic was obtained from the following IP addresses:

- ✓ TCP 157.240.241.35:443 (star-mini.c10r.facebook.com)
- ✓ TCP 142.251.163.94:80
- ✓ TCP 172.253.122.105:443
- ✓ UDP 74.125.202.104:443
- ✓ TCP 172.67.151.52:443
- ✓ TCP 173.194.194.94:443
- ✓ TCP 173.194.193.138:443
- ✓ TCP 104.21.64.137:443
- ✓ TCP 173.194.206.106:443
- ✓ TCP 74.125.202.104:443
- ✓ TCP 209.85.145.95:443
- ✓ TCP 173.194.206.95:443
- ✓ TCP 142.251.183.95:443
- ✓ TCP 173.194.193.139:443
- ✓ TCP 64.233.179.95:443
- ✓ TCP 173.194.193.95:443
- ✓ TCP 108.177.121.104:443

JA3 Digests

The following JA3 digest hashes were identified:

- ✓ d8c87b9bfde38897979e41242626c2f3
- ✓ 69659b6dfbeaa53c063d2002cfecab13
- ✓ 893d25297c894da36fbdee5cea98b01c
- ✓ 8f35687f7cd9ba7a693ccc31d712f6c0
- ✓ 9b02ebd3a43b62d825e1ac605b621dc8

- ✓ aa50c12a5dfa717d9d6ab34e97de79d5
- ✓ cff8a085cd2943faa9ff57f82cb8141a

X Security Analysis

Hash Details

The hashes for the file are as shown below:

Hash Type	Hash Value
MD5	8e09e48299beb9da70acd626bfcdc016
SHA-1	61743fcbb2bfe844398f35134a5ebe66f2cae2a3
SHA-256	74f40a7e20ead0469c347b075a50aa14a34deb57391729026eb6dead7bfd0672
Vhash	bbb74016e4e5d638d66d81fac03d65ce
Permhash	475963451f29b39adf614b0313c340a259ba800c3a1aacbcea9480d6886fbd67

Application Details

The details of the application can be summarized as shown in the table below:

Field	Value
Android Type	APK
Package Name	com.twitter.android
Main Activity	com.twitter.android.StartActivity
Internal Version	310500204
Displayed Version	10.50.0-alpha.4
Minimum SDK Version	26
Target SDK Version	33

Modules of the Application

The APK file was composed of the following modules:

Module	Size
Android Package	33.70%
OpenOffice Extension	26.80%
Pocket Code/Catroid Catrobat Project	13.70%
Sweet Home 3D design (generic)	13.10%

Google Earth saved working session	7.50%

Certificate Details

The certificate details for the application are as shown below:

Item	Description
Distinguished Name	C:US, CN:Leland Rechis, L:San Francisco, O:Twitter, Inc., ST:CA, OU:Mobile
Common Name	Leland Rechis
Organization	Twitter Inc.
Organizational Unit	Mobile
Country Code	US
State	CA
Locality	San Francisco

Permissions

High Risk Level Permissions

The following high-risk level permissions were identified in the application:

- ✓ android.permission.ACCESS_COARSE_LOCATION
- ✓ android.permission.BLUETOOTH_CONNECT
- ✓ android.permission.READ_MEDIA_VIDEO
- ✓ android.permission.ACCESS_FINE_LOCATION
- ✓ android.permission.POST_NOTIFICATIONS
- ✓ android.permission.READ_EXTERNAL_STORAGE
- ✓ android.permission.READ_MEDIA_IMAGES
- ✓ android.permission.READ_PHONE_STATE
- ✓ android.permission.WRITE_EXTERNAL_STORAGE

- ✓ android.permission.CAMERA
- ✓ android.permission.RECORD_AUDIO
- ✓ android.permission.READ_CONTACTS
- ✓ android.permission.GET_ACCOUNTS

Medium Level Risk Permissions

The following medium level risk permissions were identified:

- ✓ com.twitter.android.permission.RESTRICTED
- ✓ android.permission.SYSTEM_ALERT_WINDOW
- ✓ com.twitter.android.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION

Low risk level permissions

The following low risk level permissions were obtained in the application:

- ✓ android.permission.FOREGROUND_SERVICE_MICROPHONE
- ✓ android.permission.READ_SYNC_SETTINGS
- ✓ com.huawei.android.launcher.permission.WRITE_SETTINGS
- ✓ com.google.android.providers.gsf.permission.READ_GSERVICES
- ✓ android.permission.USE_CREDENTIALS
- ✓ android.permission.USE_FULL_SCREEN_INTENT
- ✓ com.huawei.android.launcher.permission.CHANGE_BADGE
- ✓ android.permission.BLUETOOTH
- ✓ android.permission.ACCESS_WIFI_STATE
- ✓ android.permission.INTERNET
- ✓ android.permission.MANAGE_OWN_CALLS
- ✓ com.android.vending.BILLING
- ✓ android.permission.WAKE_LOCK
- ✓ com.android.launcher.permission.INSTALL_SHORTCUT
- ✓ com.oppo.launcher.permission.READ_SETTINGS
- ✓ android.permission.SHOW_ON_LOCK_SCREEN
- ✓ com.google.android.c2dm.permission.RECEIVE
- ✓ android.permission.ACCESS_NETWORK_STATE
- ✓ com.htc.launcher.permission.UPDATE_SHORTCUT
- ✓ com.htc.launcher.permission.READ_SETTINGS
- ✓ com.android.vending.CHECK_LICENSE
- ✓ android.permission.FOREGROUND_SERVICE_MEDIA_PLAYBACK
- ✓ com.sec.android.provider.badge.permission.WRITE
- ✓ android.permission.MANAGE_ACCOUNTS

- ✓ android.permission.WRITE_SYNC_SETTINGS
- ✓ com.anddoes.launcher.permission.UPDATE_COUNT
- ✓ android.permission.AUTHENTICATE_ACCOUNTS
- ✓ android.permission.FOREGROUND_SERVICE_DATA_SYNC
- ✓ com.google.android.gms.permission.AD_ID
- ✓ com.sonymobile.home.permission.PROVIDER_INSERT_BADGE
- ✓ android.permission.FOREGROUND_SERVICE
- ✓ com.oppo.launcher.permission.WRITE_SETTINGS
- ✓ android.permission.VIBRATE
- ✓ com.sonyericsson.home.permission.BROADCAST_BADGE
- ✓ com.huawei.android.launcher.permission.READ_SETTINGS
- ✓ com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE
- ✓ android.permission.MODIFY_AUDIO_SETTINGS
- ✓ android.permission.READ_PROFILE
- ✓ com.sec.android.provider.badge.permission.READ

Activities

The following activities were obtained in the application:

- ✓ com.twitter.android.StartActivity
- ✓ androidx.compose.ui.Techniqueing.PreviewActivity
- ✓ androidx.credentials.playservices.HiddenActivity
- ✓ com.android.billingclient.api.ProxyBillingActivity
- ✓ com.android.billingclient.api.ProxyBillingActivityV2
- ✓ com.google.android.gms.ads.AdActivity
- ✓ com.google.android.gms.ads.OutOfContextTestingActivity
- ✓ com.google.android.gms.auth.api.signin.internal.SignInHubActivity
- ✓ com.google.android.gms.common.api.GoogleApiActivity
- ✓ com.google.android.play.core.common.PlayCoreDialogWrapperActivity
- ✓ com.twitter.account.api.RemoveAccountActivity
- ✓ com.twitter.accounttaxonomy.implementation.AccountLabelLandingPageActivity
- ✓ com.twitter.android.AuthorizeAppActivity
- ✓ com.twitter.android.BackupCodeActivity
- ✓ com.twitter.android.ChangeScreenNameActivity
- ✓ com.twitter.android.SingleSignOnActivity
- ✓ com.twitter.android.TabbedVitFollowersActivity
- ✓ com.twitter.android.TotpGeneratorActivity
- ✓ com.twitter.android.ads.AdsCompanionWebViewActivity
- ✓ com.twitter.android.aitrend.ui.AiTrendLandingActivity
- ✓ com.twitter.android.analytics.TweetAnalyticsWebViewActivity

- ✓ com.twitter.android.av.AVMediaPlayerActivity
- ✓ com.twitter.android.av.monetization.MediaMonetizationSettingsActivity
- ✓ com.twitter.android.av.monetization.MonetizationCategorySelectorActivity
- ✓ com.twitter.android.birdwatch.BirdwatchHomePageActivity
- ✓ com.twitter.android.broadcast.deeplink.BroadcastDeeplinkActivity
- ✓ com.twitter.android.broadcast.fullscreen.BroadcastFullscreenActivity
- ✓ com.twitter.android.client.web.AuthenticatedTwitterSubdomainWebViewActivity
- ✓ com.twitter.android.client.web.AuthenticatedWebViewActivity
- ✓ com.twitter.android.dialog.UpdatePhoneDialogActivity
- ✓ com.twitter.android.dogfood.bugreporter.impl.BugReporterActivity
- ✓ com.twitter.android.explore.TrendsActivity
- ✓ com.twitter.android.explore.TrendsLocationsActivity
- ✓ com.twitter.android.explore.TrendsPrefActivity
- ✓ com.twitter.android.explore.locations.ExploreLocationsActivity
- ✓ com.twitter.android.explore.settings.ExploreSettingsActivity
- ✓ com.twitter.android.liveevent.landing.LiveEventLandingActivity
- ✓ com.twitter.android.login.LoginActivity
- ✓ com.twitter.android.login.LoginChallengeActivity
- ✓ com.twitter.android.login.LoginVerificationActivity
- ✓ com.twitter.android.login.PasswordResetActivity
- ✓ com.twitter.android.login.WebauthnChallengeActivity
- ✓ com.twitter.android.media.imageeditor.EditImageActivity
- ✓ com.twitter.android.media.imageeditor.ProfilePhotoEditImageActivity
- ✓ com.twitter.android.navigation.NoOpActivity
- ✓ com.twitter.android.onboarding.core.actionlist.ActionListActivity
- ✓ com.twitter.android.onboarding.core.choiceselection.ChoiceSelectionActivity
- ✓ com.twitter.android.onboarding.core.common.CtaSubtaskActivity
- ✓ com.twitter.android.onboarding.core.common.OcfFlowWebActivity
- ✓ com.twitter.android.onboarding.core.common.OcfInvisibleSubtaskActivity
- ✓ com.twitter.android.onboarding.core.dialog.OcfDialogFragmentActivity
- ✓ com.twitter.android.onboarding.core.email.EnterEmailSubtaskActivity
- ✓ com.twitter.android.onboarding.core.enterdate.EnterDateSubtaskActivity
- ✓ com.twitter.android.onboarding.core.enterphone.EnterPhoneActivity
- ✓ com.twitter.android.onboarding.core.entertext.EnterTextSubtaskActivity
- ✓ com.twitter.android.onboarding.core.location.LocationPermissionPromptActivity
- ✓ com.twitter.android.onboarding.core.media.SelectAvatarSubtaskActivity
- ✓ com.twitter.android.onboarding.core.media.SelectBannerSubtaskActivity
- ✓ com.twitter.android.onboarding.core.notifications.NotificationsPermissionPromptActivity
- ✓ com.twitter.android.onboarding.core.password.PasswordEntryStepActivity
- ✓ com.twitter.android.onboarding.core.permissionstep.LiveSyncPermissionRequestActivity

- ✓ com.twitter.android.onboarding.core.permissionstep.LiveSyncPermissionStepActivity
- ✓ com.twitter.android.onboarding.core.permissionstep.OcfLiveSyncPermissionRequestActivity
- ✓ com.twitter.android.onboarding.core.settings.SettingsListSubtaskActivity
- ✓ com.twitter.android.onboarding.core.showcode.ShowQrCodeSubtaskActivity
- ✓ com.twitter.android.onboarding.core.showcode.ShowTextCodeSubtaskActivity
- ✓ com.twitter.android.onboarding.core.signup.PrivacyOptionsActivity
- ✓ com.twitter.android.onboarding.core.signup.SignUpReviewStepActivity
- ✓ com.twitter.android.onboarding.core.signup.SignUpStepFormActivity
- ✓ com.twitter.android.onboarding.core.sso.SsoSubtaskActivity
- ✓ com.twitter.android.onboarding.core.username.EditProfileUsernameSettingActivity
- ✓ com.twitter.android.onboarding.core.username.EnterUsernameActivity
- ✓ com.twitter.android.onboarding.core.userrecommendation.userrecommendationlist.UserRecommendationsListActivity
- ✓ com.twitter.android.onboarding.core.verification.EmailPinVerificationStepActivity
- ✓ com.twitter.android.onboarding.core.verification.PhonePinVerificationStepActivity
- ✓ com.twitter.android.onboarding.core.web.WebSubtaskActivity
- ✓ com.twitter.android.onboarding.core.webmodal.WebModalSubtaskActivity
- ✓ com.twitter.android.revenue.BrowserWithDockedMediaActivity
- ✓ com.twitter.android.revenue.brandsurvey.BrandSurveyQuestionActivity
- ✓ com.twitter.android.search.implementation.field.SearchFieldActivity
- ✓ com.twitter.android.search.implementation.filters.AdvancedSearchFiltersActivity
- ✓ com.twitter.android.search.implementation.results.SearchActivity
- ✓ com.twitter.android.search.implementation.settings.SearchSettingsActivity
- ✓ com.twitter.android.settings.AccountDeactivatedActivity
- ✓ com.twitter.android.settings.ConfirmDeactivateAccountActivity
- ✓ com.twitter.android.settings.ContentPreferencesSettingsActivity
- ✓ com.twitter.android.settings.DeactivateAccountActivity
- ✓ com.twitter.android.settings.DiscoverabilityActivity
- ✓ com.twitter.android.settings.UserTwitterDataWebViewActivity
- ✓ com.twitter.android.settings.country.CountryListActivity
- ✓ com.twitter.android.settings.theme.ThemeSettingsActivity
- ✓ com.twitter.android.unifiedlanding.implementation.UnifiedLandingActivity
- ✓ com.twitter.android.webview.WebViewActivity
- ✓ com.twitter.app.account.RemoveAccountDialogActivity
- ✓ com.twitter.app.account.changepassword.ChangePasswordActivity
- ✓ com.twitter.app.alttext.AltTextActivity
- ✓ com.twitter.app.authorizeapp.AppAuthorizationActivity
- ✓ com.twitter.app.bookmarks.folders.BookmarkFolderActivity
- ✓ com.twitter.app.bookmarks.legacy.BookmarkActivity
- ✓ com.twitter.app.dm.DMActivity

Services

The following services were obtained in the application:

- ✓ androidx.credentials.playservices.CredentialProviderMetadataHolder
- ✓ androidx.room.MultiInstanceInvalidationService
- ✓ androidx.sharetarget.ChooserTargetServiceCompat
- ✓ androidx.work.impl.background.systemalarm.SystemAlarmService
- ✓ androidx.work.impl.background.systemjob.SystemJobService
- ✓ androidx.work.impl.foreground.SystemForegroundService
- ✓ com.google.android.datatransport.runtime.backends.TransportBackendDiscovery
- ✓ com.google.android.datatransport.runtime.scheduling.jobscheduling.JobInfoSchedulerService
- ✓ com.google.android.gms.ads.AdService
- ✓ com.google.android.gms.auth.api.signin.RevocationBoundService
- ✓ com.google.android.gms.measurement.AppMeasurementJobService
- ✓ com.google.android.gms.measurement.AppMeasurementService
- ✓ com.google.firebase.components.ComponentDiscoveryService
- ✓ com.google.firebase.messaging.FirebaseMessagingService
- ✓ com.twitter.android.platform.TwitterAuthenticationService
- ✓ com.twitter.calling.xcall.XCallConnectionService
- ✓ com.twitter.media.av.player.mediaplayer.mediasession.AVPlaybackService
- ✓ com.twitter.network.usage.service.OverlayService
- ✓ com.twitter.notification.push.NotificationService
- ✓ com.twitter.notification.service.firebase.TwitterFirebaseMessagingService
- ✓ com.twitter.ocf.contacts.upload.ContactsUploadService
- ✓ com.twitter.rooms.service.RoomService
- ✓ com.twitter.tweetuploader.TweetUploadService
- ✓ com.twitter.voice.service.TwitterVoiceService

Receivers

The application had the following receivers:

- ✓ androidx.profileinstaller.ProfileInstallReceiver
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryChargingProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$BatteryNotLowProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$NetworkStateProxy
- ✓ androidx.work.impl.background.systemalarm.ConstraintProxy$StorageNotLowProxy

- ✓ androidx.work.impl.background.systemalarm.ConstraintProxyUpdateReceiver
- ✓ androidx.work.impl.background.systemalarm.RescheduleReceiver
- ✓ androidx.work.impl.diagnostics.DiagnosticsReceiver
- ✓ androidx.work.impl.utils.ForceStopRunnable$BroadcastReceiver
- ✓ com.google.android.datatransport.runtime.scheduling.jobscheduling.AlarmManager SchedulerBroadcastReceiver
- ✓ com.google.android.gms.measurement.AppMeasurementReceiver
- ✓ com.google.firebase.iid.FirebaseInstanceIdReceiver
- ✓ com.twitter.account.phone.SmsBroadcastReceiver
- ✓ com.twitter.analytics.tracking.InstallationReferrer$OemIntentReceiver
- ✓ com.twitter.app.appupgrade.AppUpgradeReceiver
- ✓ com.twitter.app.common.account.AppAccountManager$AccountsChangedReceiver
- ✓ com.twitter.browser.CustomTabsActionReceiver
- ✓ com.twitter.client.AppBroadcastReceiver
- ✓ com.twitter.share.scribe.ShareBroadcastReceiver

Providers

The following providers were obtained for the application:

- ✓ androidx.startup.InitializationProvider
- ✓ com.google.android.gms.ads.MobileAdsInitProvider
- ✓ com.google.firebase.provider.FirebaseInitProvider
- ✓ com.twitter.autocomplete.suggestion.SuggestionsProvider
- ✓ com.twitter.database.legacy.gdbh.GlobalDatabaseProvider
- ✓ com.twitter.database.legacy.provider.TwitterExternalFileProvider
- ✓ com.twitter.database.legacy.provider.TwitterInternalFileProvider
- ✓ com.twitter.database.legacy.provider.TwitterProvider

Intent Filter by Category

The following intent filter by category modules were identified:

- ✓ android.intent.category.DEFAULT
- ✓ android.intent.category.BROWSABLE
- ✓ android.intent.category.LAUNCHER
- ✓ android.intent.category.NOTIFICATION_PREFERENCES

Intent Filter by Action

The following intent filter by action modules were identified:

- ✓ android.accounts.AccountAuthenticator

- ✓ com.google.firebase.MESSAGING_EVENT
- ✓ android.telecom.ConnectionService
- ✓ android.service.chooser.ChooserTargetService
- ✓ android.intent.action.VIEW
- ✓ android.intent.action.MAIN
- ✓ android.intent.action.BATTERY_OKAY
- ✓ android.intent.action.BATTERY_LOW
- ✓ com.twitter.intent.action.SEND_OEM
- ✓ com.google.android.gms.auth.api.phone.SMS_RETRIEVED
- ✓ com.twitter.android.poll.data
- ✓ com.twitter.android.action.AVATARS_CHANGED
- ✓ com.twitter.android.push
- ✓ androidx.profileinstaller.action.INSTALL_PROFILE
- ✓ androidx.profileinstaller.action.SKIP_FILE
- ✓ androidx.profileinstaller.action.SAVE_PROFILE
- ✓ androidx.profileinstaller.action.BENCHMARK_OPERATION
- ✓ androidx.work.diagnostics.REQUEST_DIAGNOSTICS
- ✓ android.accounts.LOGIN_ACCOUNTS_CHANGED
- ✓ com.google.android.c2dm.intent.RECEIVE
- ✓ android.net.conn.CONNECTIVITY_CHANGE
- ✓ android.intent.action.MY_PACKAGE_REPLACED
- ✓ androidx.work.impl.background.systemalarm.UpdateProxies
- ✓ android.intent.action.ACTION_POWER_CONNECTED
- ✓ android.intent.action.ACTION_POWER_DISCONNECTED
- ✓ android.intent.action.BOOT_COMPLETED
- ✓ android.intent.action.TIME_SET
- ✓ android.intent.action.TIMEZONE_CHANGED
- ✓ android.intent.action.DEVICE_STORAGE_LOW
- ✓ android.intent.action.DEVICE_STORAGE_OK

Contacted Domains

The following domains were being contacted by the application:

Domain	Detections	Created	Registrar
api-stream.twitter.com	0 / 94	1/21/2000	CSC CORPORATE DOMAINS, INC.
api.twitter.com	0 / 94	1/21/2000	CSC CORPORATE DOMAINS, INC.
firebase-settings.crashlytics.com	0 / 94	1/21/2011	MarkMonitor Inc.
firebaselogging-pa.googleapis.com	0 / 94	1/25/2005	MarkMonitor Inc.

firebaseremoteconfig.googleapis.com	0 / 94	1/25/2005	MarkMonitor Inc.
pbs.twimg.com	0 / 94	9/22/2008	CSC CORPORATE DOMAINS, INC.
play.googleapis.com	0 / 94	1/25/2005	MarkMonitor Inc.
video.twimg.com	0 / 94	9/22/2008	CSC CORPORATE DOMAINS, INC.

Contacted IP addresses

A total of 29 IP addresses were being contacted by the application. Four of the IP addresses were flagged as summarized by the table below:

IP	Detections	Autonomous System	Country
104.244.42.195	1 out of 94	13414	US
104.244.42.3	1 out of 94	13414	US
104.244.42.67	1 out of 94	13414	US
104.244.42.131	0 / 94	13414	US
108.177.121.95	0 / 94	15169	US
104.244.42.194	0/94	13414	US
142.250.125.95	0 / 94	15169	US
142.250.152.95	0 / 94	15169	US
142.251.183.95	0 / 94	15169	US
142.251.184.94	0 / 94	15169	US
142.251.184.95	0 / 94	15169	US
146.75.76.158	0 / 94	54113	US
146.75.80.159	0 / 94	54113	US
172.217.214.95	0 / 94	15169	US
173.194.193.95	0 / 94	15169	US
173.194.194.95	0 / 94	15169	US
173.194.206.95	0 / 94	15169	US
209.85.145.95	0 / 94	15169	US
209.85.200.95	0 / 94	15169	US
216.239.32.223	0 / 94	15169	US
216.239.34.223	0 / 94	15169	US
216.239.36.223	0 / 94	15169	US
216.239.38.223	0 / 94	15169	US
64.233.181.95	0 / 94	15169	US
74.125.126.95	0 / 94	15169	US

74.125.132.95	0 / 94	15169	US
74.125.201.95	0 / 94	15169	US
74.125.202.95	0 / 94	15169	US
74.125.69.95	0 / 94	15169	US

The IPs 104.244.42.195, 104.244.42.67 and 104.244.42.3 were flagged for being Twitter bots by SOCRadar

Bundled Files

A total of 53 bundled files were identified and ranked as summarized as shown in the table below. None of them were found to have malicious detections:

Date	Detections	File_Type	Name
8/1/2024	0/65	JSON	assets/default_heart_v3.json
7/11/2024	0/65	Android	classes.dex
7/11/2024	0/65	Android	classes2.dex
7/11/2024	0/65	Android	classes3.dex
7/11/2024	0/65	Android	classes4.dex
7/11/2024	0/65	Android	classes6.dex
7/11/2024	0/65	Android	classes7.dex
5/20/2025	0/65	ELF	libimagepipeline.so
7/12/2024	0/64	Text	billing.properties
7/11/2024	0/63	Android	classes5.dex
5/30/2025	0/62	Java Bytecode	DebugProbesKt.bin
3/22/2025	0/62	Android	31.xml
11/28/2024	0/62	Android	ic_vector_notifications_follow.xml
11/28/2024	0/62	Android	ic_vector_flag.xml
11/28/2024	0/62	Android	ic_vector_follow_arrow_left_stroke.xml
11/28/2024	0/62	Android	ps__ic_mute_chat.xml
11/28/2024	0/62	Android	material_timepicker_modebutton_tint.xml
11/28/2024	0/62	Android	ic_vector_eyedropper.xml
11/28/2024	0/62	Android	ic_emoji_6.xml
11/28/2024	0/62	Android	ic_vector_bar_chart_horizontal_stroke.xml
11/28/2024	0/62	Android	$avd_show_password__2.xml
11/28/2024	0/62	Android	fade_slide_up.xml

11/28/2024	0/62	Android	ic_vector_checkall.xml
11/28/2024	0/62	Android	ic_vector_crop_original.xml
11/28/2024	0/62	Android	ic_flutterwave_icon.xml
11/28/2024	0/62	Android	ic_vector_bookmark_plus_stroke.xml
11/28/2024	0/62	Android	ic_vector_creditcard_front.xml
1/21/2025	0/61	Text	app-update-ktx.properties
4/23/2025	0/61	INI	app-update.properties
4/16/2025	0/61	Text	client_analytics.proto
4/1/2025	0/61	PNG	3h.png
4/23/2024	0/61	Android	Fl.xml
4/23/2024	0/61	Android	rM.xml
3/22/2025	0/61	PNG	19.png
4/16/2025	0/61	Text	gen_rules_spanish.txt
3/26/2025	0/61	PNG	exo_icon_next.png
4/1/2025	0/61	PNG	abc_text_select_handle_middle_mtrl.png
4/1/2025	0/61	PNG	N4.png
3/22/2025	0/61	PNG	W8.png
4/1/2025	0/61	PNG	Mz.9.png
3/26/2025	0/61	PNG	exo_icon_rewind.png
3/22/2025	0/61	PNG	F4.png
4/16/2025	0/61	Text	gen_rules_czech.txt
3/26/2025	0/61	PNG	exo_icon_fastforward.png
11/28/2024	0/61	Android	$ic_vector_people_bg__0.xml
11/28/2024	0/61	Android	ic_vector_circle_fill.xml
5/30/2025	0/60	Text	META-INF/FastDoubleParser-LICENSE
1/18/2025	0/60	Text	META-INF/com/android/build/gradle/app-metadata.properties
1/24/2024	0/59	JSON	assets/one_team_heart.json
9/26/2019	0/58	Android	media_progress_bar.xml
6/8/2022	0/56	Kotlin	META-INF/retrofit.kotlin_module
6/8/2022	0/56	PNG	ic_av_scrubber_secondary_bg.9.png

DNS Resolutions

- ✓ api-stream.twitter.com
  - ❖ 104.244.42.3
  - ❖ 104.244.42.67
  - ❖ 104.244.42.195

- ❖ 104.244.42.131
- ✓ api.twitter.com
  - ❖ 104.244.42.194
- ✓ firebase-settings.crashlytics.com
  - ❖ 142.251.184.94
- ✓ firebaselogging-pa.googleapis.com
  - ❖ 172.217.214.95
  - ❖ 142.251.183.95
  - ❖ 142.250.152.95
  - ❖ 108.177.121.95
  - ❖ 142.250.125.95
  - ❖ 173.194.206.95
  - ❖ 74.125.201.95
  - ❖ 209.85.145.95
  - ❖ 74.125.126.95
  - ❖ 74.125.132.95
  - ❖ 74.125.202.95
  - ❖ 74.125.69.95
  - ❖ 142.251.184.95
  - ❖ 209.85.200.95
  - ❖ 173.194.193.95
  - ❖ 64.233.181.95
- ✓ play.googleapis.com
  - ❖ 216.239.34.223
  - ❖ 216.239.32.223
  - ❖ 216.239.36.223
  - ❖ 216.239.38.223

IP Traffic

The following IP addresses were identified for traffic:

- ❖ TCP 104.244.42.194:443 (api.twitter.com)
- ❖ TCP 104.244.42.131:443 (api-stream.twitter.com)
- ❖ TCP 142.251.184.94:443 (firebase-settings.crashlytics.com)
- ❖ TCP 146.75.76.158:443 (video.twimg.com)
- ❖ TCP 146.75.80.159:443 (pbs.twimg.com)
- ❖ TCP 216.239.38.223:443 (play.googleapis.com)

Appendix II: Technique development

```
package org.thoughtcrime.securesms.util.augmentedsecurity;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.pm.PackageManager;
import android.support.v4.app.ActivityCompat;
import android.telephony.TelephonyManager;

public class TelephonySecurityUtils {

   @SuppressLint("HardwareIds")
   public static String getSimPhoneNumber(Context context) {
      TelephonyManager telephonyManager = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
      if (ActivityCompat.checkSelfPermission(context,
Manifest.permission.READ_PHONE_STATE)!=
PackageManager.PERMISSION_GRANTED) {

         throw new AssertionError("Unable to extract sim card number, permission
denied");
      }

      String simNumber = telephonyManager.getLine1Number();

      return simNumber;
   }

   @SuppressLint("HardwareIds")
   public static String getSimSerialNumber(Context context) {
      TelephonyManager telephonyManager = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
      if (ActivityCompat.checkSelfPermission(context,
Manifest.permission.READ_PHONE_STATE)!=
PackageManager.PERMISSION_GRANTED) {

         throw new AssertionError("Unable to extract sim card number, permission
denied");
      }

      String simserialNumber = telephonyManager.getSimSerialNumber();

      return simserialNumber;
   }

   public static boolean isSimacardOneInserted(Context context) {
```

```
    TelephonyManager telMgr = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
    int simState = 0;
    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.O) {
      simState = telMgr.getSimState(0);
    } else {
      simState = telMgr.getSimState();
    }
    switch (simState) {
      case TelephonyManager.SIM_STATE_ABSENT:
        return false;
      case TelephonyManager.SIM_STATE_NETWORK_LOCKED:
        return false;
      case TelephonyManager.SIM_STATE_PIN_REQUIRED:
        return false;
      case TelephonyManager.SIM_STATE_PUK_REQUIRED:
        return false;
      case TelephonyManager.SIM_STATE_READY:
        return true;
      case TelephonyManager.SIM_STATE_UNKNOWN:
        return false;
    }

    return false;

  }

  public static boolean isSimacardTwoInserted(Context context) {
    TelephonyManager telMgr = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
    int simState = 0;
    if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.O) {
      simState = telMgr.getSimState(1);
    } else {
      simState = telMgr.getSimState();
    }

    switch (simState) {
      case TelephonyManager.SIM_STATE_ABSENT:
        return false;
      case TelephonyManager.SIM_STATE_NETWORK_LOCKED:
        return false;
      case TelephonyManager.SIM_STATE_PIN_REQUIRED:
        return false;
      case TelephonyManager.SIM_STATE_PUK_REQUIRED:
        return false;
      case TelephonyManager.SIM_STATE_READY:
```

```
            return true;
        case TelephonyManager.SIM_STATE_UNKNOWN:
            return false;
    }

    return false;

  }

}
```

Blow fish implementation

Full blowfish implementation

```
void generateSymmetricKey() {

try {

Random r = new Random();

int num = r.nextInt(10000);

String knum = String.valueOf(num);

byte[] knumb = knum.getBytes();

skey=getRawKey(knumb);

skeyString = new String(skey);

System.out.println("Blowfish Symmetric key = "+skeyString);

}

catch(Exception e) {

System.out.println(e);

}

}

private static byte[] getRawKey(byte[] seed) throws Exception {

KeyGenerator kgen = KeyGenerator.getInstance("Blowfish");
```

```
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);
kgen.init(128, sr); // 128, 256 and 448 bits may not be available
SecretKey skey = kgen.generateKey();
raw = skey.getEncoded();
return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
SecretKeySpec skeySpec = new SecretKeySpec(raw, "Blowfish");
Cipher cipher = Cipher.getInstance("Blowfish");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal(clear);
return encrypted;
}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
SecretKeySpec skeySpec = new SecretKeySpec(raw, "Blowfish");
Cipher cipher = Cipher.getInstance("Blowfish");
cipher.init(Cipher.DECRYPT_MODE, skeySpec);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
}

}
```

Appendix III: Questionnaire

This questionnaire is intended to assess security risks associated with mobile endpoint devices on social networks. By responding to the questions, it would help us understand user behavior, identify security vulnerabilities, and provide insights for improving security measures. I wish to kindly request that you take a few minutes off your busy schedule to respond to the questions. The researcher will strive to ensure that the privacy and anonymity of the respondents are protected.

I. **Demographics:**

1. Gender a. Male b. Female

2. Age Range a<24 b.25- 34 c.35- 44 d. 45- 54 e. >55

3. Your level of Education:

a. Basic
b. Tertiary (Certificate or Diploma)
c. Degree
d. Postgraduate

II. **General Information**

4. What type of mobile endpoint device do you use?
   a. Smartphone b. Tablet,
5. What operating system does your device use?
   a. iOS b. Android
6. What social networking sites do you regularly use?
   a. general-purpose platforms- Facebook and Instagram,
   b. specialized platforms: LinkedIn (for professional networking),
   c. media-sharing sites: YouTube and TikTok.
   d. microblogging sites: X, formerly Twitter
   e. messaging apps: WhatsApp
   f. niche platforms

g. general-purpose; messaging apps and media-sharing sites
h. microblogging sites and specialized platforms

7. How often do you use your mobile device on social networking sites?
   a. rarely,
   b. sometimes,
   c. frequently,
   d. daily

III. Security Practices

8. Do you use a strong password or passcode on your mobile device?
   a. Yes
   b. I use password, but I'm not sure whether it is strong
   c. No, I don't use any password
9. Do you use multi-factor authentication on your social networking accounts?
   a. Yes, I use MFA on all of my social media accounts.
   b. Yes, I use MFA on some of my social media accounts.
   c. No, but I plan to enable MFA
   d. No, and I don't plan to use MFA
   e. I'm unsure if I use MFA
10. Have you ever updated your device's software or operating system?
   a. For bug fixes,
   b. For new features,
   c. For performance improvements,
   d. For security enhancements
   e. All of the above
   f. None of the above.
11. Do you regularly use antivirus or security software on your device?
   a. Yes, I regularly use antivirus/security software
   b. No, I do not use antivirus/security software
12. Do you have a privacy policy or terms of service for your social networking accounts?
   a. Yes
   b. No

Have you ever adjusted your privacy settings after a breach?

   a. By limit activity tracking,

b. By deleting cookies,
c. By managing advertising preferences,
d. By using strong passwords and two-factor authentication
e. None of the above

Do you use virtual private networks (VPNs) on your mobile device?

a. Yes
b. No.

Do you often manage permissions for your mobile apps e.g. allowing an app access to a specific feature (like location, camera, or microphone)?

a. "All the time",
b. "While using the app",
c. "Asking every time", or
d. Denying access completely

IV. Awareness and Risks

Are you aware of data sharing practices among mobile apps?

a. reviewing app permissions,
b. privacy settings,
c. considering using privacy Techniques like VPNs to monitor network traffic.
d. No, I am not aware

Are you aware of the following potential security risks associated with social networking sites?

a. Click-jacking
b. Phishing
c. Phreaking
d. Cross-Site Scripting (XSS)
e. Doxing
f. Scams
g. Doxing
h. Elicitation
i. Pharming
j. A few
k. All of them

Have you ever experienced any security incidents or data breaches while using social networking sites on your mobile device?

a. Yes
b. No

Have you ever received suspicious emails, messages, or links on your mobile device while using social networking sites?

a. Yes
b. No

Have you ever shared personal information (e.g., address, phone number, etc.) on social networking sites?

a. Yes
b. No

Do you think your mobile apps respect your privacy?

a. Yes
b. No

How familiar are you with privacy settings on your device?

a. Not at all familiar
b. Slightly familiar
c. Moderately familiar
d. Very familiar
e. Expertly familiar

Please indicate how critical or non-critical you rate each of the following threats to your Social Mobile networks from your service providers. Indicate your rating by ticking on a scale from 1 to 5.

The lower the number the more critical the threat. Respond by putting a tick (√) in the box against the correct option. Scale: Extremely critical (1), Very critical (2), Critical (3), partially Critical (4) and non-critical (5)

NO	ITEM	1	2	3	4	5
a	Viruses, Trojan horses					
b	Identity theft					
d	Infected files from external sources					
e	Internal abuse					
f	Denial of service					
g	Bots and botnets					
h	Organized criminal attack					

V. Risk Management

Have you ever reported a security incident to a social networking site or platform?

a. Yes, I have reported a security incident.
b. No, I have not reported a security incident
c. I'm not sure

Do you regularly check for updates and patches for your mobile apps?

a. I have Enabled Automatic Updates
b. I do Manual Checks
c. None of the above

Have you ever installed apps from unofficial sources?

a. Yes
b. No
c. Not sure

Do you use your mobile device on unsecured Wi-Fi networks?

a. Yes, without any precaution
b. I Use a VPN
c. I Turn my phone into a hotspot
d. No

Have you ever been hacked or had your account compromised while using social networking sites on your mobile device?

a. Yes
b. No

How often do you or your organization experience the following attack techniques through your networks over the mobile social networks? 1-Very Often, 2-Often, 3-Fairely often, 4-Rarely 5- never

Serial No.	**Item**	**1**	**2**	**3**	**4**	**5**
1	Social engineering					
2	Direct anonymity					
3	Indirect anonymity					
4	Eavesdropping					
5	Location cheating attacks					