

An Empirical Security Assessment of Android-Based Social Media Applications in Thailand: A Multi-Method Approach Aligned with OWASP Mobile Top 10 (2024)

Chatphat Titiakarawongse¹, Benjamas Rueangphimai¹, Chanikan Weangnon¹

¹Institute of Digital Arts and Science, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand

Correspondence: Chatphat Titiakarawongse, Institute of Digital Arts and Science, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand. Email: chatphat@sut.ac.th

Received: May 19, 2026

Accepted: July 7, 2026

Online Published: July 13, 2026

doi:10.11114/smc.v14i4.8850

URL: <https://doi.org/10.11114/smc.v14i4.8850>

Abstract

Mobile social media applications are integral to communication and content sharing but remain highly susceptible to security and privacy threats. Whilst prior research has investigated mobile vulnerabilities, few studies have systematically analysed Android-based social media applications within regional contexts such as Thailand, where usage is extensive. This study addresses this gap through a multi-tool security assessment of eight widely used Android applications. Using MobSF, the AndroBugs Framework, and reverse engineering tools (APKTool, Dex2jar, JD-GUI), vulnerabilities were identified and categorised according to the OWASP Mobile Top 10 (2024) security categories. Results revealed that M9 (Insecure Data Storage) was present in all applications, whereas M6 (Inadequate Privacy Controls) was least frequent, and no occurrences of M1 (Improper Credential Usage) or M2 (Inadequate Supply Chain Security) were observed. X (formerly Twitter) exhibited the highest vulnerability count, whilst TikTok showed the fewest. The findings highlight the value of a multi-tool approach for comprehensive vulnerability detection and underscore the urgent need for improved data handling, cryptographic safeguards, and continuous security testing in mobile application development.

Keywords: Android application security, vulnerability assessment, OWASP Mobile Top 10 (2024) security categories, social media applications, mobile cybersecurity

1. Introduction

Mobile social media applications have become integral to the contemporary digital ecosystem, fundamentally transforming communication, commerce, and social interaction, particularly in Thailand where adoption continues to expand at scale. As of 2024, 68.3% of the Thai population actively uses social media, while Android devices account for 74.8% of the mobile operating system market, with platforms such as Facebook, Instagram, TikTok, and LINE collectively reaching over 49 million users (Belson, 2024; Nukulomprattana, 2024). This widespread adoption increases cybersecurity risks by exposing mobile applications to threats such as data breaches, credential harvesting, insecure Application Programming Interface (API) exploitation, malware injection, insecure data storage, weak cryptographic implementation, and certificate validation failures. Recent high-profile security incidents involving Facebook and TikTok further emphasise the urgent need to strengthen mobile application security (Lakshmanan, 2024; S, 2024).

Although previous studies have investigated Android security using static, dynamic, and hybrid approaches, important gaps remain, including limited context-specific analyses, overreliance on automated tools, and insufficient integration of complementary assessment techniques aligned with the OWASP Mobile Top 10 (2024) security categories (Archibong et al., 2024; Cong et al., 2024; Hayes et al., 2020; Khan et al., 2024; Senanayake et al., 2023; Tang et al., 2020). To address these gaps, this study conducts a structured exploratory security assessment of eight widely used Android-based social media applications in Thailand—LINE, Discord, Pinterest, Instagram, X (formerly Twitter), Facebook, Messenger, and TikTok. The study is guided by four research questions: (RQ1) What are the prevalent security vulnerabilities in these applications? (RQ2) To what extent does integrating automated static analysis with reverse engineering improve the detection of visible and latent vulnerabilities? (RQ3) How are the identified vulnerabilities distributed across the OWASP Mobile Top 10 (2024) security categories? (RQ4) What mitigation strategies can strengthen the security posture of future application releases? To answer these questions, the study employs a multi-method framework that combines automated static analysis using MobSF and the AndroBugs Framework with reverse engineering through APKTool, Dex2jar, and

JD-GUI. By integrating complementary security assessment techniques, providing context-specific empirical evidence from Thailand, and systematically mapping identified vulnerabilities to the OWASP Mobile Top 10 (2024) security categories, this study contributes to both the mobile application security literature and evidence-based cybersecurity practice, while offering practical recommendations for developers, security practitioners, and policymakers.

2. Background and Related Work

Mobile application security, particularly on the Android platform, has become a major research focus due to the increasing prevalence of security vulnerabilities. Among the available security assessment frameworks, the OWASP Mobile Top 10 (2024) security categories are widely recognised for identifying and classifying key mobile application security risks. Accordingly, previous studies have extensively adopted the OWASP Mobile Top 10 (2024) security categories alongside tools such as the Mobile Security Framework (MobSF) and the AndroBugs Framework (Abdullah & Zeebaree, 2021; Anwar et al., 2023; Archibong et al., 2024; Chen et al., 2022; Hassan et al., 2022; Kadir et al., 2025; Keteku et al., 2024; Palanisamy et al., 2024; Taylor & Martinovic, 2017; Titiakarawongse et al., 2024; Tsobdjou et al., 2024). Building on this body of work, this section reviews the evolution of Android vulnerability assessment, introduces the OWASP Mobile Top 10 (2024) security categories, and synthesises related studies that inform the methodological design of the present study.

2.1 Mobile Application Vulnerability Analysis

Security threats targeting Android applications have become increasingly frequent and sophisticated, exposing users to risks such as data leakage, credential compromise, privilege escalation, and device takeover (Zimperium, 2024). Consequently, Android security assessment primarily relies on two complementary approaches: Static Application Security Testing (SAST), which analyses application code without execution, and Dynamic Application Security Testing (DAST), which evaluates application behaviour during runtime (Li, 2020; Zakeya et al., 2022).

SAST effectively identifies vulnerabilities such as insecure configurations, hardcoded credentials, weak cryptographic implementation, insecure data storage, and permission misuse. Its scalability and reproducibility make it well suited for large-scale exploratory studies. In contrast, DAST detects runtime vulnerabilities, including insecure API communication, authentication flaws, and abnormal application behaviour, but generally requires greater resources and depends on runtime environments and application configurations (Li, 2020; Zakeya et al., 2022).

Previous studies demonstrate that integrating complementary assessment techniques improves vulnerability detection. Khan et al. (2024) combined static and dynamic analysis using MobSF to identify hardcoded credentials, cryptographic weaknesses, and certificate validation issues, whereas Myat and Kyaw (2019) showed that reverse engineering revealed malicious logic overlooked by automated tools (Khan et al., 2024; Myat & Kyaw, 2019). Similarly, Hayes et al. (2020) reported that integrating multiple analysis techniques uncovered privacy-related behaviours beyond those identified through permission analysis alone (Hayes et al., 2020). Despite these advances, limited research has systematically evaluated the complementary value of integrating automated static analysis with structured reverse engineering within a context-specific mobile application ecosystem. This gap provides the methodological rationale for the multi-method approach adopted in the present study.

2.2 The OWASP Mobile Top 10 (2024) Security Categories

The OWASP Mobile Top 10 (2024) security categories provide an internationally recognised framework for identifying, classifying, and mitigating critical mobile application security risks, enabling consistent vulnerability assessment and comparison across studies (OWASP, 2024). The framework comprises ten security categories: M1 (Improper Credential Usage), M2 (Inadequate Supply Chain Security), M3 (Insecure Authentication/Authorization), M4 (Insufficient Input/Output Validation), M5 (Insecure Communication), M6 (Inadequate Privacy Controls), M7 (Insufficient Binary Protections), M8 (Security Misconfiguration), M9 (Insecure Data Storage), and M10 (Insufficient Cryptography). Collectively, these categories provide a comprehensive basis for evaluating the security posture of mobile applications that process sensitive user information.

Accordingly, this study adopts the OWASP Mobile Top 10 (2024) security categories as the primary analytical framework for systematically identifying, classifying, and interpreting security vulnerabilities in eight widely used Android-based social media applications in Thailand (OWASP, 2024). Using a standardised taxonomy facilitates consistent vulnerability classification and supports meaningful comparison of security findings across applications.

2.3 Related Work: Empirical and Comparative Evidence

Previous studies have consistently reported recurring security vulnerabilities across Android applications, although their methodological approaches and application contexts vary. Automated security analyses using MobSF have identified vulnerabilities including cryptographic weaknesses, certificate validation issues, insecure data storage, and security misconfigurations, demonstrating the value of systematic vulnerability assessment throughout the software development lifecycle (Khan et al., 2024; Paramasivam et al., 2024). Complementary reverse engineering studies have further revealed

excessive permissions, malicious logic, WebView vulnerabilities, insecure storage practices, API misuse, input validation weaknesses, and inadequate encryption that may not be fully detected through automated analysis alone (Abdullah & Zeebaree, 2021; Almomani & Khayer, 2020; Kim, 2020; Myat & Kyaw, 2019; Shezan et al., 2017).

More recent research has emphasised the importance of aligning vulnerability assessment with the OWASP Mobile Top 10 (2024) security categories to improve consistency in vulnerability classification and risk evaluation (Kumar, 2024; Tsobdjou et al., 2024). Other studies have extended this perspective by examining privacy-invasive data handling and Artificial Intelligence (AI)-driven security mechanisms in mobile and social media environments (Ahmed & Rashid, 2024; Hayes et al., 2020). Despite these advances, existing evidence is primarily derived from general Android applications or regional case studies in countries such as Myanmar, Bangladesh, Malaysia, and Canada, with limited attention to widely used Android-based social media applications in Thailand. Moreover, few studies have systematically integrated automated static analysis with structured reverse engineering while comprehensively mapping identified vulnerabilities to the OWASP Mobile Top 10 (2024) security categories. These contextual and methodological gaps provide the primary motivation for the present study.

2.4 Mapping Research Questions to Variables and Methods

To ensure methodological transparency and theoretical consistency, each research question is systematically mapped to its corresponding variables, analytical methods, and supporting literature. RQ1 examines security vulnerabilities using the OWASP Mobile Top 10 (2024) security categories through static analysis and reverse engineering, whereas RQ2 evaluates the complementary value of integrating automated and manual analysis techniques. RQ3 analyses the distribution of identified vulnerabilities across OWASP categories, and RQ4 synthesises mitigation strategies based on established secure software development practices. This mapping is informed by previous studies on Android security assessment and mobile application protection (Abdullah & Zeebaree, 2021; Alhamed et al., 2023; Almomani & Khayer, 2020; Hayes et al., 2020; Khan et al., 2024; Kim, 2020; Kumar, 2024; Myat & Kyaw, 2019; Paramasivam et al., 2024; Shezan et al., 2017; Tsobdjou et al., 2024).

Building on this mapping, the study adopts a hybrid analytical framework that integrates MobSF, the AndroBugs Framework, and structured reverse engineering using APKTool, Dex2jar, and JD-GUI. Identified vulnerabilities are systematically mapped to the OWASP Mobile Top 10 (2024) security categories to support consistent comparison across applications. The following section describes the research design, sampling strategy, variables, analytical procedures, and study limitations.

3. Materials and Methods

3.1 Research Design

This study employed a quantitative descriptive–exploratory research design, complemented by a systems-thinking perspective, to assess and compare security vulnerabilities across eight widely used Android-based social media applications in Thailand. Vulnerabilities were systematically identified, classified, and compared using the OWASP Mobile Top 10 (2024) security categories as the primary analytical framework (OWASP, 2024). The research combined structured static security analysis, reverse engineering, and comparative cross-application analysis to provide a consistent and systematic evaluation. Data collection and analysis were conducted between August 2024 and February 2025 using a standardised assessment procedure across all applications.

In this study, a *vulnerability* refers to an identifiable security weakness, insecure configuration, implementation flaw, or exposure risk detected through static analysis and reverse engineering, rather than a confirmed exploitable vulnerability. Accordingly, no dynamic testing, penetration testing, or live exploitation was performed, and all analyses were conducted solely for security assessment purposes in accordance with ethical and legal research guidelines.

3.2 Sample Selection and Contextual Justification

A purposive sampling strategy was employed to ensure contextual relevance to Thailand’s digital ecosystem. Eight Android-based social media applications—Facebook, LINE, TikTok, Messenger, Instagram, X (formerly Twitter), Pinterest, and Discord—were selected based on high market penetration and active user engagement in Thailand (Nukulomprattana, 2024), extensive processing of personal and sensitive user data, documented security incident histories, and diversity in functionality and user demographics. Collectively, these applications represent a broad range of communication paradigms, including messaging services, multimedia sharing platforms, microblogging networks, and community-based environments, enabling a comparative security analysis.

From a security perspective, the selected applications were further justified by their large-scale handling of sensitive data, including communication metadata, chat histories, location information, financial transaction records (e.g., marketplace activities and LINE Pay), multimedia content and metadata, and behavioural profiling data used for advertising and recommendation systems. Reported security incidents, including data exposure, account hijacking, and access control vulnerabilities, reinforce their suitability as high-impact case studies.

The analysed versions correspond to those available during the data collection period (August 2024–February 2025), ensuring temporal consistency in vulnerability assessment despite minor version traceability constraints. Usage statistics indicate highest adoption for Facebook (v474.0.0.0.95; 126.7 MB) and LINE (v6.20; 12.2 MB) at 91.5% and 90.5%, respectively, followed by TikTok (v36.3.3; 376.6 MB) at 83.0% and Messenger (v480.0.0.0.67; 63 MB) at 81.5%. Instagram (v347.0.0.0.85; 67.4 MB) shows moderate penetration at 64.5%, while X (v10.50.0-alpha.4; 120.6 MB), Pinterest (v12.32.0; 80.1 MB), and Discord (v237.13 Stable; 283.5 MB) represent lower usage at 49.6%, 26.7%, and 10.8%, respectively.

WhatsApp and WeChat were excluded due to methodological constraints, specifically strong end-to-end encryption limiting static vulnerability analysis (WhatsApp) and comparatively lower market penetration in Thailand (WeChat). Overall, this sampling design ensures representation across high-, moderate-, and niche-usage platforms, supporting robust cross-application and cross-severity comparative security analysis.

3.3 Analytical Tools and Technical Environment

To address limitations identified in prior single-method studies (Albakri et al., 2022; Cong et al., 2024; Kim, 2020; Kumar, 2024; Mente & Bagadi, 2017; Myat & Kyaw, 2019; Paramasivam et al., 2024; Senanayake et al., 2023; Shezan et al., 2017; Tang et al., 2020), this study employed a hybrid static analysis approach combining automated vulnerability scanning with structured reverse engineering to improve detection accuracy and analytical depth.

Automated static analysis was first conducted using the Mobile Security Framework (MobSF) (Abraham, 2023; Alqaradaghi et al., 2022; Anwar et al., 2023; Khan et al., 2024; Paramasivam et al., 2024) and the AndroBugs Framework (Lin, 2015, 2023). These tools were used to identify vulnerabilities related to insecure configurations, permission misuse, cryptographic weaknesses, exposed APIs, hardcoded secrets, and manifest-level security issues. All detected findings were systematically classified according to the OWASP Mobile Top 10 (2024) security categories (OWASP, 2024) to ensure standardised vulnerability categorisation and cross-application comparability.

To complement automated analysis and reduce false negatives, a reverse engineering pipeline was then applied using APKTool, Dex2jar, and JD-GUI. APKTool was used to decompile Android Package Kit (APK) files and extract AndroidManifest.xml, resources, and Smali code; Dex2jar enabled conversion of Dalvik bytecode into Java-compatible format; and JD-GUI supported high-level source code inspection. This process facilitated deeper examination of hardcoded credentials, obfuscated logic, insecure API calls, improper cryptographic implementations, and hidden execution paths that may not be detectable through automated tools alone.

Outputs from automated scanning and reverse engineering were cross-validated to remove redundancies and reduce false positives, thereby improving reliability and analytical robustness (Albakri et al., 2022; Cong et al., 2024; Kim, 2020; Kumar, 2024; Myat & Kyaw, 2019; Paramasivam et al., 2024; Senanayake et al., 2023; Shezan et al., 2017; Tang et al., 2020). Confirmed vulnerabilities were mapped to OWASP Mobile Top 10 (2024) security categories (OWASP, 2024) and evaluated in terms of exploitability, technical severity, and impact on Confidentiality, Integrity, and Availability (CIA Triad). Applications were subsequently assessed along a relative risk continuum, and mitigation insights were derived based on identified vulnerability patterns.

Overall, the integration of automated and manual reverse engineering techniques strengthens internal validity, improves detection coverage, and enhances reproducibility, despite inherent limitations such as tool dependency and obfuscation constraints.

3.4 Measurement, Risk Assessment, and Data Analysis Framework

To enhance methodological transparency and reproducibility, this study operationalised key variables and established a structured risk assessment and data analysis framework prior to evaluation. The primary dependent variable, vulnerability count, was defined as the total number of confirmed security weaknesses per application following cross-validation between automated tools and reverse engineering outputs. All vulnerabilities were mapped to the OWASP Mobile Top 10 (2024) security categories (M1–M10) (OWASP, 2024) to ensure standardised classification and cross-application comparability.

Independent variables included permission count, defined as the total number of permissions declared in AndroidManifest.xml, and over-privileged permissions, identified through comparison between declared permissions and observed application behaviour during reverse engineering. Additional security indicators included hardcoded secrets (e.g., credentials, API keys, tokens), insecure configurations (e.g., improper Transport Layer Security (TLS) implementation, exposed endpoints, debug flags), and cryptographic weaknesses (e.g., deprecated or misconfigured algorithms). These variables enabled both quantitative measurement and qualitative interpretation of recurring security issues such as over-permissioning, insecure authentication handling, and privacy exposure risks (Cong et al., 2024; Kim, 2020; Kumar, 2024; Paramasivam et al., 2024; Senanayake et al., 2023; Shezan et al., 2017; Tang et al., 2020).

Severity assessment followed a multi-factor model combining tool-generated risk indicators from MobSF and AndroBugs with the OWASP Mobile Top 10 (2024) security categories criticality benchmarks (OWASP, 2024). Each vulnerability was further evaluated based on exploitability (attack complexity, required attacker capability, and exposure level) and impact on confidentiality, integrity, and availability. Based on this framework, vulnerabilities were categorised into low, moderate, and severe severity levels, representing increasing levels of exploitability and potential system impact.

Data analysis combined quantitative and qualitative techniques. Quantitative analysis included frequency distribution across the OWASP Mobile Top 10 (2024) security categories and comparative ranking of applications based on total and severity-weighted vulnerability scores. A severity-weighted model ensured that high-impact vulnerabilities contributed proportionally to overall risk profiling. Qualitative thematic analysis of reverse engineering outputs was used to identify recurring weaknesses, including insecure design patterns, implementation flaws, and architectural vulnerabilities in Android-based social media applications.

Overall, integrating automated detection with manual validation strengthened internal validity, reduced tool dependency bias, and improved the reliability, transparency, and reproducibility of the analytical framework.

3.5 Ethical Compliance

All analyses were conducted under strict ethical and legal constraints to ensure responsible cybersecurity research practice. The study was limited to non-intrusive static analysis of APKs legally obtained from official sources, without performing any live exploitation, penetration testing, network interaction, or runtime manipulation. All assessments were confined to offline inspection of application components, including manifest files, configurations, and decompiled code, and all identified vulnerabilities were analysed solely for academic and defensive purposes to enhance security awareness and mitigation practices. This approach ensures full compliance with ethical standards and reinforces responsible vulnerability research without risking service disruption or user data compromise.

3.6 Methodological Limitations

Although the proposed hybrid static analysis framework enhanced detection depth, cross-validation reliability, and methodological rigor, several inherent limitations remain. First, the study primarily relied on SAST and did not incorporate DAST, runtime analysis, or penetration testing. Consequently, runtime-dependent vulnerabilities—such as session hijacking, race conditions, dynamic code loading and execution, memory corruption, and context-dependent execution flaws—could not be empirically observed or validated. Accordingly, outputs generated by tools such as MobSF, the AndroBugs Framework, APKTool, Dex2jar, and JD-GUI should be interpreted as static indicators of potential vulnerabilities rather than confirmed evidence of runtime exploitability (Cong et al., 2024; Kim, 2020; Kumar, 2024; Li, 2020; Senanayake et al., 2023; Tang et al., 2020).

Second, the scope of the study is limited to eight Android-based social media applications in the Thai digital context. This relatively narrow and context-specific sample constrains the generalisability of the findings to other application domains, platforms, and geographic regions. As a result, the observed vulnerability patterns should be interpreted as context-bound rather than universally representative of the broader mobile application ecosystem.

Third, the study did not include live exploitation, penetration testing, runtime instrumentation, or network-level traffic analysis due to ethical, legal, and operational constraints associated with evaluating widely deployed commercial applications. Consequently, vulnerabilities related to authentication flows, session management, secure communication channels, token handling mechanisms, transport-layer security, insecure API interactions, and data storage practices were assessed exclusively through static evidence. In this context, findings classified according to the OWASP Mobile Top 10 (2024) security categories, particularly M9 (Insecure Data Storage), should be interpreted as indicative risk signals rather than confirmed exploit conditions (OWASP, 2024). Furthermore, backend services, cloud infrastructure configurations, and server-side API behaviours remain outside the analytical boundary of APK-level inspection.

Fourth, reverse engineering outputs are subject to technical limitations including code obfuscation, incomplete decompilation, and semantic reconstruction loss associated with tools such as Dex2jar and JD-GUI. Although manual Smali-level inspection and cross-validation procedures were applied, certain execution paths, reflective calls, dynamically loaded components, and deeply obfuscated logic structures may not have been fully reconstructed. Additionally, AndroidManifest.xml analysis was limited to declarative configuration inspection, exported component analysis, and permission mapping, without formal modelling of inter-component communication or permission dependency relationships. These constraints are consistent with prior studies on Android reverse engineering limitations (Albakri et al., 2022; Myat & Kyaw, 2019; Senanayake et al., 2023; Tang et al., 2020).

Finally, the study did not employ quantitative evaluation metrics, such as precision, recall, F1-score, and false-positive and false-negative rates, to assess tool performance. Moreover, the proposed framework was not benchmarked against established security assurance frameworks, including the Open Worldwide Application Security Project Mobile

Application Security Verification Standard (OWASP MASVS), the National Institute of Standards and Technology Secure Software Development Framework (NIST SSDF), the Common Weakness Enumeration (CWE) classification, and Software Composition Analysis (SCA)-based evaluation models. This limits direct comparability with formally validated security assessment frameworks, although it does not compromise internal methodological consistency (Cong et al., 2024; Kumar, 2024; Li, 2020; Paramasivam et al., 2024; Senanayake et al., 2023; Tang et al., 2020).

Overall, while the hybrid approach strengthens analytical coverage and internal validity, future research should extend the methodology to include DAST, runtime instrumentation, controlled penetration testing, and standardized benchmarking frameworks to improve empirical validation, external validity, and generalisability of findings.

4. Results and Discussion

This study presents a security assessment of eight widely used Android-based social media applications in Thailand using MobSF and the AndroBugs Framework, complemented by reverse engineering techniques (APKTool, Dex2jar, and JD-GUI). The integrated approach enables the identification of both surface-level and deeper vulnerabilities, revealing patterns in vulnerability distribution, severity, and application-specific risks. The results and discussion are structured according to the research questions, covering methodological validation, vulnerability and permission analysis, OWASP category mapping, tool comparison, and mitigation implications.

4.1 Reliability and Justification of Methods

The reliability of the findings is supported by a systematic and rigorously validated methodological design that integrates automated static analysis with manual reverse engineering to capture both structural and implementation-level vulnerabilities. MobSF and the AndroBugs Framework ensure consistent detection coverage and reproducibility, while manual inspection of Smali and decompiled Java code enables result verification, reduction of false positives, and identification of hidden vulnerabilities, thereby strengthening internal validity in line with established mobile security evaluation practices (Albakri et al., 2022; Cong et al., 2024; Halani et al., 2021; Myat & Kyaw, 2019; Senanayake et al., 2023; Tang et al., 2020; Thant & Tin, 2023).

Reliability is further reinforced through a replicable workflow, including APK acquisition from official sources within a defined collection period, consistent tool configuration, and systematic cross-validation between automated outputs and manual analysis. In addition, the purposive selection of eight Android applications—based on high adoption in Thailand, extensive handling of sensitive data, and documented security concerns—supports contextual relevance and practical representativeness. Severity assessment further integrates tool-generated risk indicators with contextual evaluation of exploitability and user impact, enhancing analytical consistency.

Collectively, these procedures improve reproducibility, credibility, and methodological rigor, aligning with prior studies in mobile application security assessment (Cong et al., 2024; Kim, 2020; Kumar, 2024; Paramasivam et al., 2024; Senanayake et al., 2023; Tang et al., 2020).

4.2 Comprehensive Analysis of Vulnerabilities, Permissions, and Source Code with Occurrence Summary

The comprehensive analysis of vulnerabilities, permissions, and source code reveals key insights into the security posture of eight widely used Android-based social media applications in Thailand. Overall, M9 was identified in all applications (100%), indicating a systemic weakness in protecting data at rest, including insufficient encryption, weak file permissions, and insecure local storage practices. M3 and M8 were each observed in 87.5% of applications, reflecting widespread issues in access control, session handling, and insecure configuration settings. M5 appeared in 75% of cases, highlighting persistent risks in the protection of data in transit. Collectively, these results indicate that core security mechanisms related to data protection, authentication, and secure communication remain inconsistently implemented, increasing exposure to unauthorised access and data leakage.

In contrast, M6 was identified in 12.5% of applications, suggesting partial adoption of privacy safeguards, although residual risks remain in data minimisation and consent handling. M1 and M2 were not detected in any application, indicating relatively stronger compliance in these areas. Overall, the findings demonstrate a mixed security landscape, where improvements in selected OWASP domains coexist with persistent vulnerabilities in core security controls (OWASP, 2024).

Severity classification in this study was derived using a combined approach integrating tool-generated risk scores (MobSF and AndroBugs) with risk guidance from the OWASP Mobile Top 10 (2024) security categories. Each vulnerability was assessed based on exploitability (attack complexity, required privileges, and exposure) and potential impact on confidentiality, integrity, and availability. Based on this framework, vulnerabilities were consistently categorised into low, moderate, and severe levels: low severity indicates limited exploitability and minimal impact; moderate severity reflects conditionally exploitable weaknesses with partial data or control exposure; and severe severity represents highly exploitable vulnerabilities with significant risks of data compromise or system-level impact. This classification scheme

was applied uniformly across all tables and analyses to ensure consistency and comparability of results.

The findings summarised in the accompanying table present the frequency and severity distribution of identified vulnerabilities across all applications, supporting comparative analysis of overall security posture. In relation to **RQ1**, the results highlight M9, M3, M8, and M5 as the most prevalent and high-impact vulnerability classes (Cong et al., 2024; Kim, 2020; Kumar, 2024; Paramasivam et al., 2024; Senanayake et al., 2023; Shezan et al., 2017; Tang et al., 2020).

Table 1. Vulnerability Comparison of Eight Widely Used Android-Based Social Media Applications in Thailand Based on the OWASP Mobile Top 10 (2024) Security Categories

Vulnerability Category	Occurrence (%)	Affected Applications	Impacts (Technical/Business)
M9: Insecure Data Storage	100.0	All applications	Severe
M3: Insecure Authentication/Authorization	87.5	Line, Discord, Pinterest, Instagram, X (formerly Twitter), Facebook, Messenger	Severe
M8: Security Misconfiguration	87.5	Line, Pinterest, Instagram, X (formerly Twitter), Facebook, Messenger, TikTok	Severe
M5: Insecure Communication	75.0	Discord, Pinterest, Instagram, X (formerly Twitter), Facebook, Messenger	Moderate/Severe
M4: Insufficient Input/Output Validation	37.5	Line, Instagram, X (formerly Twitter)	Severe
M7: Insufficient Binary Protections	37.5	Line, Discord, X (formerly Twitter)	Moderate
M10: Insufficient Cryptography	25.0	Discord, X (formerly Twitter)	Severe
M6: Inadequate Privacy Controls	12.5	Line	Low/Severe

While the occurrence and severity of vulnerabilities provide an overall risk overview, certain issues, particularly M9, may vary depending on device state and usage context. On rooted devices, insecure storage can significantly increase the attack surface, whereas sandboxing mechanisms on non-rooted devices may partially mitigate exposure (Shen et al., 2016). Although the OWASP Mobile Top 10 (2024) security framework recommends controls such as encryption, secure storage, and device-binding mechanisms, these mitigations may introduce usability trade-offs, particularly during application migration or credential recovery processes (Kumar, 2024; OWASP, 2024; Paramasivam et al., 2024). These trade-offs are acknowledged but not further examined in this study, and are suggested as a direction for future research on security–usability balance.

To complement automated vulnerability detection, reverse engineering was performed using APKTool, Dex2jar, and JD-GUI, enabling detailed static inspection of application structure and code-level behaviour (Albakri et al., 2022; Myat & Kyaw, 2019). APKTool supported extraction of AndroidManifest.xml and configuration files, revealing issues such as excessive permissions, exposed components, debug flags, and insecure backup settings, corresponding primarily to the OWASP Mobile Top 10 (2024) security categories M3, M6, M7, M8, and M9. Dex2jar and JD-GUI enabled decompilation and source-level analysis, uncovering hardcoded credentials (API keys and tokens), exposed endpoints, and weaknesses in input validation, error handling, and cryptographic implementation, aligned with M4, M5, and M10.

Overall, reverse engineering results confirmed critical weaknesses including insecure configurations, sensitive data exposure, and insufficient protective controls, even under partial obfuscation constraints (Albakri et al., 2022; Myat & Kyaw, 2019). These findings indicate persistent gaps in secure coding practices and defence-in-depth implementation, consistent with prior studies (Cong et al., 2024; Kumar, 2024; Paramasivam et al., 2024; Senanayake et al., 2023; Tang et al., 2020). In line with OWASP Mobile Top 10 (2024) recommendations, strengthening security requires adoption of secure Software Development Life Cycle (SDLC) practices, regular code review, integration of automated static analysis, improved binary protection, secure configuration management, and runtime integrity verification to mitigate reverse engineering risks and unauthorised access (Albakri et al., 2022; Cong et al., 2024; Myat & Kyaw, 2019; OWASP, 2024; Senanayake et al., 2023; Tang et al., 2020).

4.3 Integrated Vulnerability Distribution and Tool Effectiveness Analysis

The integrated analysis of vulnerability distribution and tool effectiveness reveals notable differences in the security postures of eight widely used Android-based social media applications in Thailand, while also demonstrating the value of a complementary multi-tool approach. X (formerly Twitter) exhibited the highest number of vulnerabilities ($n = 7$), including insecure data storage, weak cryptographic controls, and insecure communication channels. LINE followed with six vulnerabilities, covering authentication weaknesses, input validation issues, privacy controls, misconfiguration, and insecure storage. In contrast, TikTok demonstrated the strongest security posture with only two vulnerabilities, although residual risks remain due to its large-scale deployment. Other applications, including Instagram, Messenger, Pinterest, Facebook, and Discord, showed moderate risk profiles with four to five vulnerabilities, mainly related to insecure communication, data storage, and code protection. Overall, these patterns indicate uneven security maturity across

applications and persistent weaknesses in core security controls, particularly data protection, communication security, and resistance to code tampering (Cong et al., 2024; Kim, 2020; Kumar, 2024; Paramasivam et al., 2024; Senanayake et al., 2023; Tang et al., 2020).

From a methodological perspective, the combined use of MobSF, the AndroBugs Framework, and reverse engineering tools (APKTool, Dex2jar, and JD-GUI) enabled broad coverage of OWASP Mobile Top 10 (2024) security categories, strengthening analytical depth and reliability (OWASP, 2024). MobSF was particularly effective in detecting configuration- and binary-level issues, while AndroBugs excelled in identifying misconfigurations, excessive permissions, and privacy-related vulnerabilities, including exclusive detection of M6 in one application (Abraham, 2023; Lin, 2015, 2023). Overlapping detections across M3, M5, M7, and M9 further reduced tool-specific bias and improved result robustness (Cong et al., 2024; Senanayake et al., 2023; Tang et al., 2020).

Complementary reverse engineering using APKTool, Dex2jar, and JD-GUI enabled detailed inspection of application resources, bytecode, and decompiled source code, revealing hardcoded credentials, exposed endpoints, and weaknesses in authentication and cryptographic logic (Albakri et al., 2022; Myat & Kyaw, 2019). Despite limitations from code obfuscation, this integrated approach provided a comprehensive view of structural, behavioural, and implementation-level vulnerabilities, confirming the complementary value of automated scanning and reverse engineering for mobile security analysis (Cong et al., 2024; Senanayake et al., 2023; Tang et al., 2020).

These findings directly address RQ2 by confirming the effectiveness of combining automated and manual analysis techniques. The vulnerability distribution presented in the accompanying table also informs RQ3 by mapping observed issues to OWASP Mobile Top 10 (2024) security categories (OWASP, 2024). The following section builds on these results by discussing security implications and proposing targeted mitigation strategies to enhance mobile application resilience.

Table 2. Vulnerability Comparison of Eight Widely Used Android-Based Social Media Applications in Thailand Based on the OWASP Mobile Top 10 (2024) Security Categories

The OWASP Mobile Top10 (2024) Security Category	Line	Discord	Pinterest	Instagram	X (Formerly Twitter)	Facebook	Messenger	TikTok
M1								
M2								
M3	x	#	#	#	√	x	x	
M4	√			√	x			
M5		#	#	#	x	x	x	
M6	x							
M7	√	#			#			
M8	x		x	x	x	x	x	x
M9	#	#	#	#	#	x	x	x
M10		√			√			
Total Number	6	5	4	5	7	4	4	2

Remark: x found by AndroBugs, √ found by MobSF, and # found by AndroBugs and MobSF.

4.4 Vulnerability Impacts and Mitigations in the Selected Mobile Applications

The vulnerability assessment indicates that security weaknesses across the selected mobile applications have significant implications for user data protection and application integrity, spanning multiple categories of the OWASP Mobile Top 10 (2024) security categories (OWASP, 2024; Senanayake et al., 2023). Identified issues range from M3 to M10, including unauthorised access, data leakage, privacy violations, insecure communication, and code execution risks. For example, M3 relates to authentication and authorization weaknesses, while the use of device identifiers such as ANDROID_ID and IMEI contributes to tracking risks associated with authentication-related processes, including user tracking and identity spoofing. Other categories include M4, M5, M7, M8, M9, and M10. Overall, the findings highlight persistent gaps in secure coding and defence-in-depth practices, consistent with prior studies (Cong et al., 2024; OWASP, 2010, 2024; Senanayake et al., 2023), and directly address RQ4.

To mitigate these risks, a defence-in-depth approach within the secure SDLC is essential. Key measures include enforcing strong encryption for data at rest and in transit (e.g., Advanced Encryption Standard (AES)-256 and TLS 1.3), implementing robust authentication and session management with role-based access control, and replacing persistent device identifiers with secure Universally Unique Identifier (UUID)-based mechanisms. Secure configuration practices should include restricting exposed components via AndroidManifest settings, disabling unnecessary WebView features (e.g., JavaScript), and applying code obfuscation, tamper detection, and runtime integrity checks. In addition, input validation, output encoding, and parameterised queries are critical to mitigating injection-based attacks. Continuous security assurance should be integrated into Continuous Integration (CI)/ Continuous Delivery/Deployment (CD) pipelines through automated static and dynamic analysis, supported by periodic penetration testing (Cong et al., 2024; Tang et al., 2020). Compliance with data protection regulations and regular updates to cryptographic libraries further

strengthen resilience against evolving threats (Kumar, 2024; OWASP, 2024; Paramasivam et al., 2024; Senanayake et al., 2023).

The implications of these vulnerabilities extend beyond technical risks to broader societal, financial, and organisational impacts. For users, weaknesses in authentication, communication, and storage can lead to identity theft, fraud, and privacy breaches. At the organisational level, they may result in data breaches, regulatory penalties, service disruption, and reputational damage, as evidenced by incidents such as Equifax, Sony Pictures, Heartbleed, Cambridge Analytica–Facebook exposure, Amazon Web Services (AWS) S3 misconfigurations, and the Agent Smith malware campaign. These cases highlight that mobile application security is a strategic concern rather than solely a technical issue. Therefore, embedding secure coding practices, continuous vulnerability assessment, and robust data protection mechanisms is essential to enhancing application resilience and maintaining user trust (Cong et al., 2024; Kumar, 2024; Paramasivam et al., 2024; Senanayake et al., 2023; Tang et al., 2020).

5. Conclusions and Future Works

This study presents a systematic security assessment of eight widely used Android-based social media applications in Thailand, mapped against the OWASP Mobile Top 10 (2024) security categories (OWASP, 2024). The findings reveal pervasive high- and moderate-severity vulnerabilities, particularly insecure data storage, weak authentication and authorisation, insufficient input/output validation, insecure communication channels, and security misconfigurations. These vulnerabilities expose users to privacy leakage, unauthorised access, and identity misuse, while also increasing organisational exposure to regulatory, operational, and reputational risks (OWASP, 2024).

Methodologically, the study demonstrates the effectiveness of a hybrid static analysis and reverse engineering approach. The integration of MobSF and the AndroBugs Framework with APKTool, Dex2jar, and JD-GUI enables multi-layer vulnerability detection, improves cross-validation reliability, and reduces single-tool bias (Abraham, 2023; Albakri et al., 2022; Anwar et al., 2023; Cong et al., 2024; Khan et al., 2024; Lin, 2015, 2023; Myat & Kyaw, 2019; Senanayake et al., 2023; Tang et al., 2020). The results further highlight the importance of secure coding practices, hardened configurations, strong cryptographic controls, continuous vulnerability assessment, and regulatory compliance, including the Personal Data Protection Act (PDPA), to strengthen mobile application security.

However, the findings indicate that static analysis alone is insufficient to fully capture runtime-dependent, backend, and dynamic behavioural vulnerabilities in complex mobile ecosystems.

Future research should extend this study in four main directions. First, methodological enhancement should incorporate DAST, runtime behavioural analysis, network traffic inspection, authentication-flow validation, and controlled penetration testing to capture runtime-dependent vulnerabilities such as session hijacking, race conditions, and dynamic code execution (Cong et al., 2024; Kumar, 2024; Tang et al., 2020). Second, the analytical scope should be expanded to include backend services, cloud infrastructure, and server-side API security, as APK-level analysis does not capture distributed system misconfigurations and server-side vulnerabilities. Third, future studies should strengthen benchmarking and validation through quantitative performance metrics (e.g., precision, recall, F1-score, false-positive rates, and false-negative rates) and formal evaluation against established security frameworks, including OWASP MASVS, NIST SSDF, CWE classification schemes, and SCA-based evaluation models, to improve comparability and methodological rigor (Cong et al., 2024; Kumar, 2024; Li, 2020; Paramasivam et al., 2024). Fourth, intelligent and automated security mechanisms should be explored using AI/ML-based vulnerability detection, predictive threat modelling, and real-time anomaly detection, as demonstrated by emerging frameworks such as AUSERA (Chen et al., 2022; Senanayake et al., 2023; Ullah et al., 2024).

Beyond these four directions, future work should prioritise reproducible experimental environments, exploit validation protocols, and multi-tool semantic fusion techniques to enhance reliability, external validity, and reproducibility. Security–usability trade-offs should also be further investigated, particularly for high-impact vulnerabilities such as M9, where stronger protections may introduce operational constraints in real-world deployment contexts.

Overall, this study contributes empirical evidence on the security posture of widely used mobile social media applications and highlights the need for a comprehensive, multi-layered, and continuously evolving security assessment approach to strengthen mobile ecosystem resilience.

Acknowledgments

The authors acknowledge the use of generative AI tools to assist in improving English grammar, clarity, and sentence structure during manuscript preparation. The AI tools were used solely for language editing purposes, and the authors take full responsibility for the accuracy, integrity, and originality of the work.

Author contributions

Chatphat Titiakarawongse formulated the research problem, reviewed the relevant literature, conceptualised the research idea and methodology, designed and validated the research methodology, interpreted and analysed the data, validated the research findings, visualised the results, discussed the findings, and wrote, reviewed, edited, and revised the manuscript. Benjamas Rueangphimai and Chanikan Weangnon conducted the experiments and collected the raw data. All authors read and approved the final version of the manuscript.

Funding

This research received no funding.

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Informed consent

Obtained.

Ethics approval

The Publication Ethics Committee of the Redfame Publishing.

The journal's policies adhere to the Core Practices established by the Committee on Publication Ethics (COPE).

Provenance and peer review

Not commissioned; externally double-blind peer reviewed.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

Data sharing statement

No additional data are available.

Open access

This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/4.0/>).

Copyrights

Copyright for this article is retained by the author(s), with first publication rights granted to the journal.

References

- Abdullah, H., & Zeebaree, S. R. M. (2021). Android Mobile Applications Vulnerabilities and Prevention Methods: A Review. 2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA), Baghdad, Iraq. <https://doi.org/10.1109/IT-ELA52201.2021.9773615>
- Abraham, A. (2023). *Mobile-Security-Framework-MobSF* GitHub, Inc. Retrieved October, 2024 from <https://github.com/MobSF/Mobile-Security-Framework-MobSF>
- Ahmed, L. K., & Rashid, R. D. (2024). Advanced Security: The Application Social Media and Their Security. The 2nd International Conference on Applied Computing & Smart Cities (ICACS24) Erbil, Iraq. <https://doi.org/10.1051/itmconf/20246401006>
- Albakri, A., Fatima, H., Mohammed, M., Ahmed, A., Ali, A., Ali, A., & Elzein, N. M. (2022). Survey on Reverse-Engineering Tools for Android Mobile Devices. *Mathematical Problems in Engineering*, 2022, 7. <https://doi.org/10.1155/2022/4908134>
- Alhamed, M., Bukhowah, R., Alsahaim, S., & Frikha, M. (2023). Penetration Testing in Mobile Devices, Vulnerabilities and Solutions. *Journal of Theoretical and Applied Information Technology*, 101(12), 15.
- Almomani, I. M., & Khayer, A. A. (2020). A Comprehensive Analysis of the Android Permissions System. *IEEE Access*, 8, 18. <https://doi.org/10.1109/ACCESS.2020.3041432>
- Alqaradaghi, M., Morse, G., & Kozsik, T. (2022). Detecting Security Vulnerabilities with Static Analysis – A Case Study. *Pollack Periodica: An International Journal for Engineering and Information Sciences*, 17(2), 7. <https://doi.org/10.1556/606.2021.00454>

- Anwar, C., Sumerli, C. H., Hady, S., Rahayu, N., & Kraugusteeliana, K. (2023). The Application of Mobile Security Framework (MOBSF) and Mobile Application Security Testing Guide to Ensure the Security in Mobile Commerce Applications. *Jurnal Sistim Informasi dan Teknologi*, 5(2), 97-102. <https://doi.org/10.37034/jsisfotek.v5i1.231>
- Archibong, E. E., Stephen, B. U.-A., & Asuquo, P. (2024). Analysis of Cybersecurity Vulnerabilities in Mobile Payment Applications *Archives of Advanced Engineering Science*, 1–11. <https://doi.org/10.47852/bonviewAAES42022595>
- Belson, D. (2024). *Cloudflare 2024 Year in Review*. Retrieved December, 2024 from <https://blog.cloudflare.com/radar-2024-year-in-review/>
- Chen, S., Zhang, Y., Fan, L., Li, J., & Liu, Y. (2022). AUSERA: Automated Security Vulnerability Detection for Android Apps. the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), Rochester, Michigan, the United States of America. <https://doi.org/10.1145/3551349.3559524>
- Cong, N. T., Toan, L. H., & Thanh, T. M. (2024). An Overview of Static and Dynamic Analysis in Application Security Testing. *Journal of Military Science and Technology*, 99, 11. <https://doi.org/10.54939/1859-1043.j.mst.99.2024.1-11>
- Halani, K. R., Kavita, & Saxena, R. (2021). Critical Analysis of Manual Versus Automation Testing. 2021 International Conference on Computational Performance Evaluation (ComPE), Shillong, India. <https://doi.org/10.1109/ComPE53109.2021.9752388>
- Hassan, M. A., Shukur, Z., & Mohd, M. (2022). A Penetration Testing on Malaysia Popular e-Wallets and m-Banking Apps. *International Journal of Advanced Computer Science and Applications (IJACSA)* 13(5), 692-703. <https://doi.org/10.14569/IJACSA.2022.0130580>
- Hayes, D., Cappa, F., & Le-Khac, N. A. (2020). An Effective Approach to Mobile Device Management: Security and Privacy Issues Associated with Mobile Applications. *Digital Business*, 1(1), 8. <https://doi.org/10.1016/j.digbus.2020.100001>
- Kadir, F. M., Irsan, M., & Putrada, A. G. (2025). Benchmarking Mobile Apps Security in Universities: An OWASP Mobile Top 10 Framework Perspective. *International Journal on Information and Communication Technology (IJoICT)*, 11(1), 36-49. <https://doi.org/10.21108/ijoict.v11i1.9094>
- Keteku, J., Dameh, G. O., Mante, S. A., Mensah, T. K., Amartey, S. L., & Diekuu, J.-B. (2024). Detection and Prevention of Malware in Android Mobile Devices: A Literature Review. *International Journal of Intelligence Science*, 14(4), 71-93. <https://doi.org/10.4236/ijis.2024.144005>
- Khan, S. A., Adnan, M., Ali, A., Raza, A., Ali, A., Naqvi, S. Z. H., & Hussain, T. (2024). An Android Applications Vulnerability Analysis using MobSF. 2024 International Conference on Engineering & Computing Technologies (ICECT), Islamabad, Pakistan. <https://doi.org/10.1109/ICECT61618.2024.10581312>
- Kim, H. W. (2020). A Study on the Mobile Application Security Threats and Vulnerability Analysis Cases. *International Journal of Internet, Broadcasting and Communication*, 12(4), 8. <https://doi.org/10.7236/IJIBC.2020.12.4.180>
- Kumar, R. (2024). Vulnerability Assessment of Mobile Applications. *International Journal of Innovative Science and Research Technology*, 9(8), 6. <https://doi.org/10.38124/ijisrt/IJISRT24AUG464>
- Lakshmanan, R. (2024). *Celebrity TikTok Accounts Compromised Using Zero-Click Attack via DMs*. Retrieved October, 2024 from https://thehackernews.com/2024/06/celebrity-tiktok-accounts-compromised.html?fbclid=IwY2xjawGGIaVleHRuA2FlbQIxMAABHQzXjJuYdOJm7IHOPi3XbUaZhIxV7byS5i rkrx94uhlZlQsTmvi12xnDWQ_aem_dOnCneQImrpzvKgw4sw
- Li, J. (2020). Vulnerabilities Mapping based on OWASP-SANS: A Survey for Static Application Security Testing (SAST). *Annals of Emerging Technologies in Computing (AETiC)*, 4(3), 8. <https://doi.org/10.33166/AETiC.2020.03.001>
- Lin, Y. C. (2015). *AndroBugs Framework: An Android Application Security Vulnerability Scanner* Black Hat Europe 2015, Amsterdam, Netherland.
- Lin, Y. C. (2023). *AndroBugs Framework*. GitHub, Inc. Retrieved October, 2024 from https://github.com/AndroBugs/AndroBugs_Framework.
- Mente, R., & Bagadi, A. (2017). Android Application Security. *Advances in Computational Sciences and Technology*, 10(5), 5.
- Myat, S. M., & Kyaw, M. T. (2019). Analysis of Android Applications by Using Reverse Engineering Techniques. *International Journal of Innovative Science and Research Technology*, 4(3), 8.
- Nukulomprattana, P. (2024). *The Summary of Key Statistics and Insights Regarding Thai People's Use of Digital and Social Media in 2024*. Retrieved October, 2024 from <https://www.popticles.com/insight/thailand-digital-and-social->

media-stats-2024/

- OWASP. (2010). *OWASP Secure Coding Practices-Quick Reference Guide*. The OWASP Foundation Inc. Retrieved December, 2025 from https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf
- OWASP. (2024). *Mobile Top 10 2024: Final Release Updates*. The OWASP Foundation Inc. Retrieved October, 2024 from <https://owasp.org/www-project-mobile-top-10>
- Palanisamy, R., Muthukumarappan, A., Kattubadi, J. B., Shaikh, M. T., & Maskari, A. S. M. A. (2024). Trends and Advances in Application Security Analysis: A Comprehensive Research Study. *International Research Journal of Innovations in Engineering and Technology (IRJIET)*, 8(12), 6. <https://doi.org/10.47001/IRJIET/2024.812017>
- Paramasivam, D. a. I., Ismail, N. L., & Al-Nahari, A. (2024). Static Security Analysis of Government and Non-Government Android Mobile Applications in Malaysia: A Comparative Study Using MobSF and OWASP Mobile Top 10. *International Journal of Academic Research in Business and Social Sciences* 14(10), 2640–2662. <https://doi.org/10.6007/IJARBS/v14-i10/22955>
- S, S. (2024). *TikTok Influencers, Beware: Hackers are Using Zero-Click Techniques Through Direct Messages to Target and Take Over Accounts*. Retrieved October, 2024 from https://www.antivirus.in.th/news/21151.html?fbclid=IwY2xjawGGISxleHRuA2FlbQIxMAAB-HamdaVh3mzt4O1O6-ZZdVB2MwX3N05JrAjwTCKx1-4uqsj05IuQmUulraQ_aem__HwN0kl7GBcZJaeNR_A8Fw
- Senanayake, J., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., & Piras, L. (2023). Android Source Code Vulnerability Detection: A Systematic Literature Review. *ACM Computing Surveys*, 55(9), 37. <https://doi.org/10.1145/3556974>
- Shen, Y., Evans, N., & Benameur, A. (2016). Insights into Rooted and Non-Rooted Android Mobile Devices with Behavior Analytics. The 31st Annual ACM Symposium on Applied Computing (SAC 2016), Pisa, Italy. <https://doi.org/10.1145/2851613.2851713>
- Shezan, F. H., Afroze, S. F., & Iqbal, A. (2017). Vulnerability Detection in Recent Android Apps: An Empirical Study. 2017 International Conference on Networking, Systems and Security (NSysS), Dhaka, Bangladesh. <https://doi.org/10.1109/NSysS.2017.7885802>
- Tang, J., Li, R., Wang, K., Gu, X., & Xu, Z. (2020). A Novel Hybrid Method to Analyze Security Vulnerabilities in Android Applications. *Tsinghua Science and Technology*, 25(5), 15. <https://doi.org/10.26599/TST.2019.9010067>
- Taylor, V. F., & Martinovic, I. (2017). To Update or Not to Update: Insights From a Two-Year Study of Android App Evolution. the 2017 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '17), Abu Dhabi, United Arab Emirates. <https://doi.org/10.1145/3052973.3052990>
- Thant, K. S., & Tin, H. H. K. (2023). The Impact of Manual and Automatic Testing on Software Testing Efficiency and Effectiveness. *Indian Journal of Science and Research*, 3(3), 6.
- Titiakarawongse, C., Taksin, S., Ruangsawat, J., Deeduangpan, K., & Boonkrong, S. (2024). Comparative Vulnerability Analysis of Thai and Non-Thai Mobile Banking Applications *Journal of Cybersecurity and Privacy*, 4(3), 650-662. <https://doi.org/10.3390/jcp4030031>
- Tsobjou, L. D., Pierre, S., & Quintero, A. (2024). A Framework for Security Assessment of Android Mobile Banking Applications. *Computer Networks and Communications*, 2(1), 14. <https://doi.org/10.37256/cnc.2120243929>
- Ullah, S., Li, J., Ullah, F., Chen, J., Ali, I., Khan, S.,...Leung, V. C. M. (2024). The Revolution and Vision of Explainable AI for Android Malware Detection and Protection. *Internet of Things*, 27, 36. <https://doi.org/10.1016/j.iot.2024.101320>
- Zakeya, N., Ségla, K., Chamseddine, T., & Alvine, B. B. (2022). Probing AndroVul Dataset for Studies on Android Malware Classification. *Journal of King Saud University – Computer and Information Sciences*, 34(9), 12. <https://doi.org/10.1016/j.jksuci.2021.08.033>
- Zimperium. (2024). 2024 Global Mobile Threat Report. 37. [https://lp.zimperium.com/hubfs/MAPS_MTD/REPORT/GEN/Global%20Mobile%20Threat%20Report%202024%20FINAL%20\(1\).pdf](https://lp.zimperium.com/hubfs/MAPS_MTD/REPORT/GEN/Global%20Mobile%20Threat%20Report%202024%20FINAL%20(1).pdf)