Evaluating Penetration Testing Methodologies for Mobile Applications: A Comparative Study of Android and iOS Security

Ayorinde P. Tinuoye

Department of Informatics
King's College London
London WC2R 2LS, United Kingdom

Corresponding Author:
Ayorinde P. Tinuoye
Phone: +1-346-332-3123
Email: ayorinde.tinuoye@kcl.ac.uk

Abstract

This literature review critically examines contemporary methodologies for mobile application penetration testing, with a focus on comparing the effectiveness and limitations of testing across Android and iOS platforms. Most common approaches—including Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and network-level analysis—are evaluated for their coverage and limitations in mobile-specific contexts. The review identifies a disconnect between current penetration testing tools and the proactive, secure-by-design posture promoted by standards such as OWASP MSTG and MASVS. A key theme is the divergent architecture of Android and iOS: Android's open ecosystem supports in-depth testing through root access, emulation, and instrumentation, while iOS's tightly controlled security model enhances baseline protection but significantly restricts third-party assessment. The study highlights the limitations of emulators, tool fragmentation, and the lack of platform-agnostic methodologies. Emerging opportunities such as DevSecOps integration, Mobile Device Management (MDM) frameworks, Apple's Security Research Device initiative, and evolving cross-platform tools suggest future directions for more holistic and ethical testing practices. The review concludes that hybrid, cooperative, and policy-aligned testing strategies are essential to closing current methodological gaps and advancing mobile application security.

Index Terms: penetration testing, mobile application security, Android, iOS, OWASP MASVS, mobile testing tools, DevSecOps, application vulnerabilities.

1. Introduction

Penetration testing is used to simulate the techniques and technologies utilised by malicious actors, enabling comprehensive testing of a system's security posture. This process helps identify security issues such as vulnerabilities, hardware and software misconfigurations, and network weaknesses [46]. The outcome of penetration testing provides security administrators with valuable insights into existing security flaws, allowing them to enhance the overall resilience of their systems [10].

Penetration testing can be carried out manually or by using automated tools [35]. Regardless of the method, the process typically involves gathering detailed information about the target system beforehand (reconnaissance), identifying possible vulnerabilities, attempting to exploit those weaknesses (either digitally or physically), and reporting the findings [44]. The primary objective is to uncover security flaws. Additionally, penetration tests can help evaluate an organisation's response to security incidents, measure employee awareness of security practices, and ensure adherence to security policies [25].

Confidentiality plays a critical role in penetration testing [50]. Any vulnerabilities or information uncovered during the assessment are considered highly sensitive and are kept private until all issues have been thoroughly addressed. This approach allows the organisation to take proactive measures to fix security gaps, reducing the risk of a successful attack [38].

A. Mobile Application Security

The smartphone market, which includes devices commonly used as mobile phones, is experiencing significant and sustained growth [21]. Cinar and Kara [13] mentioned that, according to the International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker, global smartphone shipments rose by 13.2% year-on-year, with the recovery from 2020 continuing into the second quarter of 2021, reaching 313.2 million units.

Cinar and Kara [13] further highlighted that a key reason behind the widespread adoption of smartphones is the increasing range of features combined with decreasing prices. Smartphones support various connectivity options such as Bluetooth, Wi-Fi, and GPS, and their ability to run third-party applications greatly enhances their functionality. These apps are primarily distributed

through official online platforms—Google Play Store for Android devices and the Apple App Store for iOS.

However, these appealing features such as regular internet access and a wide selection of apps have also attracted cybercriminals. According to data from Kaspersky Security Network, 9,599,519 instances of malware, adware, and riskware targeting mobile devices were blocked in the third quarter of 2021 [47]. Malware can infiltrate devices through various methods, which include multimedia messaging services (MMS), email, or by exploiting security flaws in networks or the devices themselves [29]. The most common method of infection occurs when users download apps that secretly contain malicious code [19].

B. OWASP Mobile Top 10

The OWASP Mobile Security Project serves as a centralised resource offering valuable insights that help mobile app developers enhance security by addressing common vulnerabilities identified in mobile applications globally [16]. The OWASP Mobile Top 10 (2024) outlines the most critical security threats facing mobile apps and provides recommended measures to reduce their likelihood and potential impact [27]. Past vulnerabilities discovered in mobile apps are compiled into a database, which is used to recognise and categorise security flaws based on attack methods and their consequences [41].

The Open Source Foundation for Application Security (OWASP) supports two main frameworks designed to standardise the process of assessing mobile app vulnerabilities, aiming to create a consistent and unified methodology: the Mobile Application Security Verification Standard (MASVS) and the Mobile Application Security Testing Guide (MASTG) are resources from OWASP focused on mobile app security [45]. MASVS defines a set of baseline security requirements that mobile apps should meet to ensure a basic level of protection. It serves as a consistent framework for evaluating app security and helps standardise assessment practices across the board [19]. OWASP MASTG delivers in-depth guidance and best practices for conducting mobile app security testing, including detailed technical testing methods [6].

iOS and Android remain the major operating systems for mobile devices [20], together accounting for over 99% of the smartphone market [28]. O'Dea's [36] research revealed that 99.42% of

smartphone devices run on either the iOS or Android operating system. Nonetheless, this growth and the excessive reliance on these devices in daily routines have exacerbated the risk to the security of private data [28].

Table 1. **OWASP Mobile Top 10 (2024)**

M1: Improper Credential Message
M2: Inadequate Supply Chain Security
M3: Insecure Authentication/Authorisation
M4: Insufficient Input/Output Validation
M5: Insecure Communication
M6: Inadequate Privacy Controls
M7: Insufficient Binary Protections
M8: Security Misconfiguration
M9: Insecure Data Storage
M10: Insufficient Cryptography

C. Mobile Penetration Testing

The digital security landscape has been significantly reshaped by the mobile technology revolution. Mobile Penetration Testing (Mobile Pen Testing) has emerged as a key aspect of cybersecurity, specifically designed to address the unique challenges associated with mobile platforms. As mobile devices have become deeply embedded in everyday life, they have also become primary targets for cyber threats, requiring targeted security measures. Mobile penetration testing is specifically designed to address the unique security challenges of mobile apps and devices [3]. It takes a deep dive into mobile app security by examining elements such as source code, data storage, authentication mechanisms, and communication channels. Additionally, it assesses how mobile apps interact with their underlying operating systems to uncover and address potential vulnerabilities [17].

Penetration testing for mobile applications has grown more challenging and costly because of factors such as varying platforms, diverse device types, different contextual event scenarios, and

the use of offloading techniques [3]. With the rapid rise in smartphone and tablet usage, the risk of security vulnerabilities has grown substantially. These devices often hold large volumes of personal and corporate information, which makes them attractive targets for cybercriminals. Mobile penetration testing is essential because it takes a proactive approach to identifying and fixing security vulnerabilities before they can be exploited. Penetration testers can identify weaknesses in mobile applications, operating systems, and the devices they operate on [38].

Traditional penetration testing usually focuses on external threats to networks and servers, while mobile penetration testing explores the unique landscape of mobile operating systems, app environments, and how they interact [44]. This focus is essential because mobile devices introduce unique security challenges, including diverse OS configurations, vulnerabilities specific to mobile platforms, and a wide variety of mobile applications. Additionally, mobile penetration testing employs distinct tools and methodologies tailored to these environments [55]. Whereas traditional penetration testing often relies on network scanning and server vulnerability assessments, mobile penetration testing employs tools and techniques tailored to mobile environments. These include reverse engineering applications, examining how apps store and transmit data, and evaluating the security of mobile APIs [44].

2. Research Methodology

This section outlines the systematic approach employed to identify, select, and analyse relevant literature on penetration testing methodologies for mobile applications. The methodology is anchored in a transparent search protocol, explicit inclusion/exclusion criteria, and a multi-phase screening process to ensure the inclusion of high-quality, peer-reviewed, and authoritative sources.

A. Research Objective

This literature review critically examines penetration testing methodologies applied to mobile applications on Android and iOS platforms, with particular attention to how platform-specific architectures and constraints shape testing practices, tool effectiveness, and security assessment outcomes. The research questions are summarised as follows:

- **RQ1:** What are the most common penetration testing methodologies used in mobile application security assessment?
- **RQ2:** How effective are the tools used in mobile application penetration testing for detecting vulnerabilities on Android and iOS platforms?
- **RQ3:** How do Android and iOS differ in their security architectures, and how do these differences affect penetration testing strategies?
- **RQ4:** What are the key limitations and opportunities for advancing penetration testing methodologies across both platforms?

B. Search Protocol

A structured search strategy was designed to retrieve comprehensive and relevant literature addressing penetration testing methodologies for mobile applications. The search was conducted across the following electronic databases:

- IEEE Xplore
- ScienceDirect (Elsevier)
- SpringerLink
- ACM Digital Library
- Google Scholar

The search period spanned 2008 to 2025, with emphasis on recent developments (2013–2025) to account for evolving mobile platforms and security practices.

The following keyword combinations were used to refine the search:

- "Mobile application security" AND "penetration testing"
- "Android security testing" AND "iOS security testing"
- "Mobile application" AND ("vulnerability assessment" OR "threat modeling")
- "Penetration testing methodologies" AND ("Android" OR "iOS")
- "Dynamic analysis" AND "static analysis" AND "mobile apps"

Searches were limited to English-language publications, and results were initially sorted by relevance and citation impact. Preference was given to studies published in peer-reviewed journals or reputable conference proceedings.

C. Selection Criteria

To ensure the quality and relevance of the literature, both inclusion and exclusion criteria were applied during the screening process:

Inclusion Criteria:

- Peer-reviewed journal articles or conference papers
- Technical reports or white papers from reputable organizations (e.g., OWASP, NIST)
- Studies explicitly focusing on penetration testing in mobile application contexts
- Comparative analyses of Android and iOS security testing tools or frameworks

Exclusion Criteria:

- Publications focused exclusively on web application security
- Non-scholarly sources such as blogs, editorial opinions, or vendor marketing material
- Studies lacking a methodological basis or empirical data

D. Screening and Selection Process

The selection process followed a multi-stage screening procedure, inspired by systematic review best practices:

1. **Initial Retrieval and De-duplication:**
   The initial search yielded 753 records. After removing 92 duplicate entries, 661 unique records remained for screening.
2. **Title and Abstract Screening:**
   These 661 records were screened based on their titles and abstracts to assess preliminary relevance. Articles that did not mention mobile security, penetration testing, or related methodologies were excluded. 219 articles passed this phase.

3. **Full-Text Assessment:**

   The full texts of these 219 studies were retrieved and evaluated in-depth using the inclusion/exclusion criteria. Studies were assessed for:

   - Alignment with research objectives
   - Methodological rigor (e.g., empirical studies, compara3ve analysis, case studies)
   - Coverage of relevant mobile platforms (Android, iOS)

   After this phase, 44 studies met all criteria and were deemed suitable for inclusion in the final review.

4. **Documentation of Exclusions:**

   A PRISMA-style flow diagram was used to document reasons for exclusion at each phase. Common reasons for exclusion were:

   - Focus on general cybersecurity without a mobile context
   - Lack of methodological detail
   - Duplicate publication of the same study in multiple venues

E. Data Extraction and Analysis

The final dataset of 44 studies was categorised thematically into four major areas:

- Penetration Testing Methodologies for Mobile Applications (addresses RQ1)
- Effectiveness of Penetration Testing Tools in Mobile Security (addresses RQ2)
- Platform Specific Considerations: Android vs. iOS (addresses RQ3)
- Limitations and Opportunities for Advancing Mobile Penetration Testing (addresses RQ4)

Of the 44 studies included in the final dataset, 31 are directly cited within the Results and Analysis section as they most clearly address the research questions; the remaining studies informed the broader analysis and contextual understanding.

A qualitative content analysis approach was applied to extract key themes, methodologies, and findings aligned with the research objectives. The extracted data was then synthesised to identify gaps, challenges, and trends in mobile application penetration testing practices.

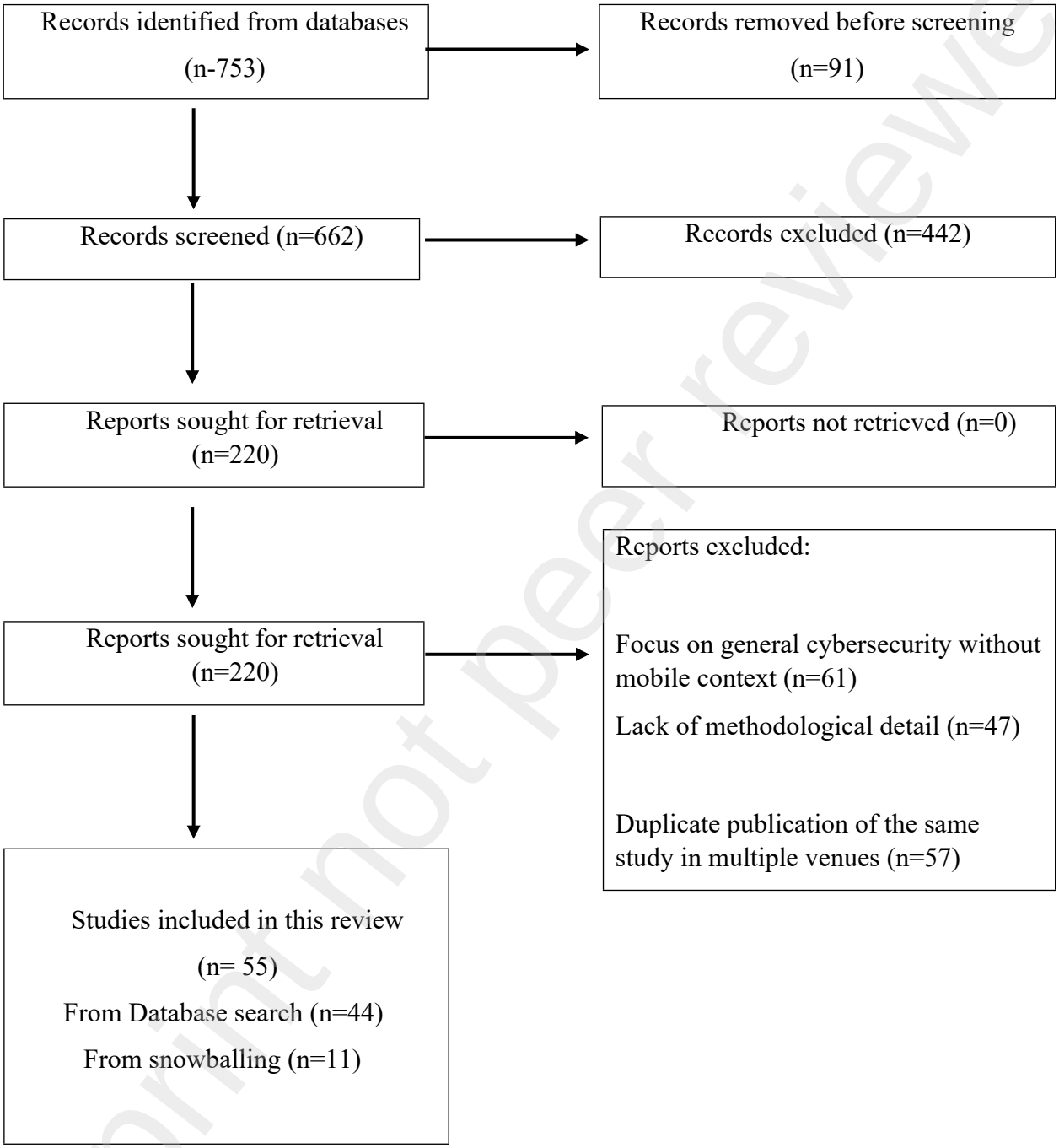
F. Supplementary Search Strategy: Snowballing

In addition to database searches, snowballing was employed to ensure the comprehensiveness of the literature review. This involved two iterative processes:

1. **Backward Snowballing**:
   References (bibliographies) of the initially selected articles were reviewed to identify earlier foundational works or relevant studies not retrieved during the keyword-based search.
2. **Forward Snowballing:**
   Citation tracking was used (via Google Scholar) to identify more recent publications that cited the initially included papers. This allowed the discovery of up-to-date research that builds upon foundational work.

The snowballing process helped identify an additional 10 studies that were not captured in the original database queries but met the inclusion criteria. These were subjected to the same screening and quality appraisal process as the initially retrieved articles.

Figure 1: PRISMA 2020 Flow Diagram for Literature Selection



3. Results and Analysis

A. Penetration Testing Methodologies for Mobile Applications

The most common penetration testing methodologies for mobile applications are Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Network-Level Analysis [37; 1]. SAST offers a proactive approach, enabling early detection of vulnerabilities by analyzing source code without executing it, which makes it particularly effective during development. However, this method falls short in identifying issues that only manifest during runtime. To address this limitation, DAST complements SAST by evaluating the application in its running state, thereby exposing vulnerabilities such as authentication flaws or improper session handling. Beyond the application layer, network-level analysis provides insight into how mobile apps communicate with external servers and services. This methodology is crucial for uncovering risks such as insecure API calls, unencrypted transmissions, session hijacking, and man-in-the-middle (MitM) vulnerabilities. Despite the strengths of these individual methods, none offer comprehensive coverage of all threat vectors. Consequently, hybrid approaches have emerged, combining static, dynamic, and network analysis to achieve broader coverage, albeit at the cost of increased complexity and resource demands [37].

B. Effectiveness of Penetration Testing Tools in Mobile Security

Mobile-specific frameworks such as the OWASP Mobile Security Testing Guide (MSTG) and the Mobile Application Security Verification Standard (MASVS) have established comprehensive criteria for evaluating mobile application security, emphasising the need to move beyond automated scanning to include manual testing of complex vulnerabilities, such as those related to authentication, data storage, and inter-process communication [5, 23]. However, there remains a significant gap between these recommended practices and the capabilities of commonly used penetration testing tools. Tools such as Metasploit, Core Impact, and OWASP ZAP, while effective in detecting known vulnerabilities, operate largely in a post development context. These tools do not facilitate secure-by-design practices, as they lack integration with early-stage development processes [22]. This disconnect suggests that current industry tools are misaligned with the proactive security posture promoted by MSTG and MASVS. The reliance on post-development fixes not only delays remediation but also increases the likelihood of costly structural

rework, underscoring the need for model-based or integrated testing approaches that align more closely with secure development life cycles.

C. Platform Specific Considerations: Android vs. iOS

Both Android and iOS implement core security features such as sandboxing, encryption, and permission models, yet their architectural choices differ substantially and have significant implications for mobile penetration testing [20, 40].

Android's open-source design offers both advantages and challenges for penetration testing. Its architecture, which includes user-level isolation via the Linux kernel and application sandboxing using unique user IDs, allows for effective separation between applications [20]. This structure not only enhances security at runtime but also enables testers to conduct static and dynamic analyses with relative ease. Tools such as emulators, debugging frameworks, and environments where root access can be obtained allow testers to manipulate app behaviour, inspect system calls, and simulate real-world attack scenarios [44]. As a result, Android offers greater accessibility for penetration testers, enabling quicker identification of security vulnerabilities [20].

By contrast, iOS operates within a closed ecosystem that employs a significantly more restrictive security model. Apple's closely coordinated hardware and software systems enforce a strong security framework featuring secure boot processes, biometric data protection via the Secure Enclave, and mandatory file encryption protocols. [14, 48]. While these measures significantly reduce the platform's attack surface, they simultaneously hinder third-party security evaluation. The inability to access source code, the difficulty of jailbreaking newer devices, and restrictions on dynamic testing through emulators or proxy tools limit testers' ability to perform thorough assessments. As noted by Norman and Norman [34], meaningful penetration testing on iOS often requires circumventing protections that Apple explicitly designs to prevent tampering.

From a practical standpoint, these differences create asymmetrical testing environments. Android allows for broader security auditing, but device fragmentation and inconsistent OEM-level security implementations can introduce unpredictability. iOS, though more uniform and secure by design, offers fewer tools for comprehensive black-box testing. This limits the ability of independent

researchers to assess the platform and shifts greater responsibility onto Apple's internal security systems and review procedures.

Moreover, the closed nature of iOS means vulnerabilities may remain obscured for longer periods unless disclosed by Apple or discovered through expensive exploit development. In enterprise contexts, this limits the options for independent verification of app security on iOS, whereas Android's openness permits more frequent and detailed third-party audits**.**

D. Limitations and Opportunities for Advancing Mobile Penetration

The evolution of mobile operating systems has produced divergent security architectures in Android and iOS, with direct consequences for penetration testing methodologies.

i. Platform Architecture and Testing Accessibility

Android's open-source architecture provides penetration testers with considerable visibility into system behaviour, source code, and debugging tools. Testers can gain root access, utilise emulators, and manipulate system internals with relative ease [[4, 54]. Tools such as Frida and Burp Suite Mobile Assistant further support dynamic analysis and runtime instrumentation [44]. This accessibility creates a testing-friendly environment but is offset by platform fragmentation: varying levels of patching, vendor-specific modifications, and inconsistent implementation of security features complicate standardised assessment [18, 52].

Conversely, iOS presents a tightly closed ecosystem with stringent access restrictions. Apple controls both hardware and software, enforces strict code-signing requirements, and employs secure boot chains, kernel integrity checks, and hardware-based cryptographic storage via the Secure Enclave [12, 14, 15]. These protections increase platform resilience but also limit the feasibility of direct penetration testing. Jailbreaking is often the only means of conducting in-depth assessments, yet it introduces legal, ethical, and practical complications [49].

ii. Methodological Limitations

Emulators and simulators serve as valuable tools for initial penetration testing, but their accuracy is limited. Android emulators might not accurately replicate custom OEM behaviors or kernel-level functions [32], whereas iOS simulators cannot reliably emulate hardware-dependent features

such as vibration and acceleration [43, 51] . These constraints limit the depth and authenticity of simulated attacks, particularly for device-specific vulnerabilities, making real devices necessary for comprehensive development and penetration testing.

Moreover, traditional penetration testing frameworks, which are designed primarily for desktop or web environments, lack native support for mobile-specific attack surfaces such as inter-app communication, intent hijacking (on Android), or URL scheme abuse (on iOS). This gap has led to the emergence of mobile-focused tools such as MobSF and Objection; however, their coverage remains inconsistent across platforms [11, 15, 42, 44].

iii. Emerging Opportunities

Despite structural limitations, several promising trends are emerging. On Android, the openness of the ecosystem allows for custom testing environments using rooted or emulated devices, supporting advanced techniques such as API hooking, runtime tampering, and syscall tracing [2, 9, 33]. The growth of automated mobile DevSecOps pipelines has also encouraged the integration of static and dynamic security tools earlier in the development lifecycle, improving vulnerability detection prior to deployment [30, 35].

In the case of iOS, recent industry and academic collaborations have introduced more ethical access routes for testing, such as security research devices offered by Apple to vetted researchers [8]. Additionally, enterprise frameworks such as Mobile Device Management (MDM) offer controlled channels to enforce and evaluate security policies without compromising system integrity [53].

Cross-platform testing tools, although still in development, offer an additional avenue of potential. Frameworks such as MobSF, when combined with automated threat modelling and code analysis, are enabling a more consistent baseline for security evaluation across both operating systems [44].

iv. Synthesis and Research Gaps

Android's openness facilitates methodological freedom at the cost of platform consistency, while iOS's uniformity ensures stronger default protections but restricts third-party analysis. A key

limitation across both platforms is the lack of comprehensive, platform-agnostic methodologies that account for unique system constraints while delivering actionable insight [44] .

Based on the literature reviewed, it appears that future advancements in mobile penetration testing will rely on hybrid models that combine automated tools, platform-specific strategies, and cooperative frameworks with OS vendors [23, 44]. Moreover, policy-level changes (such as legal protections for ethical hacking) may help reduce friction in testing iOS and proprietary builds of Android [26].

Table 2. Summary of Key Studies in Relation to Research Questions

Author(s), Year	Methodology Type	Tools / Frameworks Used	Platform(s)	Key Findings	Relevant RQ(s)	Limitations / Notes
[1, 37]	SAST, DAST, Network Analysis	Not specified	Android & iOS	Hybrid methods (SAST + DAST + network-level) offer broader coverage but increase complexity.	RQ1, RQ4	No method alone is sufficient; combining them is resource intensive.
[5, 23]	Guidelines Review, Tool Evaluation	OWASP MSTG, MASVS	Android & iOS	Emphasis on manual testing beyond automation. Tools don't align well with MSTG/MASVS secure-by-design principles.	RQ2, RQ4	Gaps exist between industry tools and best-practice frameworks.
[22]	Tool Analysis	Metasploit, Core Impact, OWASP ZAP	Android & iOS	Tools are post-development oriented; lack support for early-stage secure development practices.	RQ2, RQ4	Reactive rather than proactive; contributes to delayed remediation.
[40, 20]	Comparative Architecture Review	Not specified	Android vs. iOS	Android is more accessible for testing due to open-source nature; iOS is restrictive, impacting black-box testing.	RQ3	Android allows deeper analysis; iOS limits third-party evaluation.

[44]	Practical Tool Evaluation	Frida, Burp Suite Mobile Assistant, MobSF, Objection	Android & iOS	Android supports runtime instrumentation; iOS testing depends on workarounds. MobSF emerging as cross-platform testing tool.	RQ2, RQ3, RQ4	Tool coverage is uneven across platforms; ethical/legal hurdles with iOS.
[4]	Platform-Specific Analysis	Emulators, Rooting Tools	Android	Easy access to debugging and root features enables advanced testing.	RQ3, RQ4	Fragmentation complicates testing; not all devices behave uniformly.
[14, 48]	iOS Architecture Review	Not specified	iOS	Strong hardware/software security; impedes penetration testing.	RQ3	Closed ecosystem hinders independent evaluation.
[26]	Policy Discussion	Not applicable	Android & iOS	Legal and ethical reforms (e.g., protections for researchers) are needed to support responsible testing.	RQ4	Testing restrictions are legal as well as technical; policies matter.
[8]	Industry Initiative Report	Apple Security Research Device	iOS	Offers a legal route for researchers to test iOS devices under strict controls.	RQ4	Access is limited to vetted individuals; not universally available.

4. Discussion

Based on the analysis of the findings for the research questions in Section 2 concerning the common penetration testing methodologies used in iOS and Android applications, the effectiveness of the tools employed, the impact of differing security architectures on testing strategies, and the key limitations and opportunities for advancing mobile application penetration testing, this review presents several important insights.

In relation to the first research question, while the most common penetration testing methodologies for mobile applications fall into three categories, none can be fully relied upon in isolation. Their strengths are most effective when used in combination. This raises a critical question: if, for any reason, a penetration tester lacks the resources to implement a hybrid approach, which of the three methods is most essential for conducting a meaningful mobile application penetration test?

The answer to this question depends on context such as development stage, app type, regulations, and platform security. Early development benefits from static analysis for actionable insights with low runtime complexity, while live, interactive apps may require dynamic or network testing to reveal critical vulnerabilities. Future research should develop a context-based decision model to help testers choose the most effective method when a full hybrid approach isn't feasible.

Regarding the second question, findings reveal that no existing penetration testing tools currently support security-by-design. This does not mean they are ineffective; rather, it highlights that their utility is limited to post-development phases. This raises a critical follow-up: Are there any penetration testing tools that support security-by-design for mobile applications?

Findings for the third question suggest that iOS offers greater security than Android due to its closed security architecture, in contrast to Android's open model. However, this increased security comes at the cost of flexibility and agility. As a result, developers and testers typically have greater control on Android. Future work should examine how this affects the speed and effectiveness of vulnerability remediation.

The fourth research question highlights trade-offs between platform architecture and testing accessibility. Android's open design allows greater testing flexibility but leads to fragmentation,

complicating consistent assessments. iOS's closed system enforces uniform security but restricts external testing, often requiring jailbreaking, which raises legal and integrity concerns. Future research should examine how these structural differences affect penetration testing effectiveness.

Methodologically, both Android emulators and iOS simulators fall short of replicating real devices, limiting test accuracy. While mobile-focused tools now offer native support, they remain constrained by platform-specific limits. A cost-benefit analysis is needed on developing tools that offer consistent coverage across platforms.

Tools like MobSF show promise in enabling platform-agnostic testing. Future studies should explore their impact on detection speed and remediation efficiency.

5. Conclusion

This literature review highlights the methodological divergence and platform-specific constraints that define mobile application penetration testing. Core techniques such as SAST, DAST, and network analysis provide essential, though individually insufficient, coverage of the mobile threat landscape. While Android's open architecture facilitates deep inspection through rooting, emulation, and dynamic instrumentation, it is hindered by fragmentation and uneven security implementation. Conversely, iOS offers a more secure and uniform environment by design; yet this same rigidity significantly restricts external testing capabilities, often requiring the circumvention of built-in protections.

Current industry tools fall short of bridging this divide, with many lacking alignment with secure-by-design principles or adequate support for mobile-specific attack surfaces. Despite these challenges, emerging opportunities, which include DevSecOps integration, ethical access routes such as Apple's Security Research Device programme, and cross-platform tools such as MobSF suggest a shift towards more cohesive and adaptable testing frameworks. Ultimately, penetration testing requires the development of hybrid methodologies that integrate static, dynamic, and contextual analysis, while remaining sensitive to the operational realities of each platform. Future research should focus on establishing such standardised, platform-agnostic models and advocating for policy-level reforms to support transparent, ethical testing across mobile ecosystems.

References

1. Adamski, J., Janiszewski, M., & Rytel, M. (2025). IoT Mobile Applications

2. Agman, Y., & Hendler, D. (2021). BPFroid: robust real time Android malware detection framework. *arXiv preprint arXiv:2105.14344*.

3. Alhamed, M., & Rahman, M. H. (2023). A systematic literature review on penetration testing in networks: future research directions. *Applied Sciences*, *13*(12), 6986.

4. Almusallam, A. (2018). Penetration Testing for Android Applications with Santoku Linux.

5. Antonishyn, M. (2020). Mobile applications vulnerabilities testing model. *Collection" Information Technology and Security"*, *8*(1), 49-57.

6. Anwar, C., 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*, 97-102.

7. Al Shebli, H. M. Z., & Beheshti, B. D. (2018, May). A study on penetration testing process and tools. In *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (pp. 1-7). IEEE.

8. Apple. (2024, June 15). *Security Research Device Program overview*. Apple Inc. https://security.apple.com/research-device

9. Backes, M., Bugiel, S., Schranz, O., von Styp-Rekowsky, P., & Weisgerber, S. (2017, April). Artist: The android runtime instrumentation and security toolkit. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 481-495). IEEE.

10. Baluni, S., Dutt, S., Dabral, P., Maji, S., Kumar, A., & Chaudhary, A. (2022, October). Penetration Testing on Virtual Machines. In *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 1-6). IEEE.

11. Bhandari, S., Jaballah, W. B., Jain, V., Laxmi, V., Zemmari, A., Gaur, M. S., ... & Conti, M. (2017). Android inter-app communication threats and detection techniques. *Computers & Security*, *70*, 392-421.

12. Caruso, A. (2024). *Forensic Analysis of Mobile Spyware: Investigating Security, Vulnerabilities, and Detection Challenges in Android and iOS Platforms* (Doctoral dissertation, Politecnico di Torino).

13. Cinar, A. C., & Kara, T. B. (2023). The current state and future of mobile security in the light of the recent mobile security threat reports. *Multimedia Tools and Applications*, *82*(13), 2026920281.

14. Deaconescu, R., Enck, W., Chiroiu, M., & Deshotels, L. (2021). iOS Security Framework: Understanding the Security of Mobile Phone Platforms. In *Encyclopedia of Cryptography, Security and Privacy* (pp. 1-5). Berlin, Heidelberg: Springer Berlin Heidelberg.

15. Deshotels, L. A. (2018). *Automated Evaluation of Access Control in the iPhone Operating System*. North Carolina State University.

16. Dhingra, Y., Ranga, V., & Vishwakarma, D. K. (2024, December). A Study of the Security and Privacy Risks in Health-Related Mobile Applications in India. In *2024 International Conference on Communication, Control, and Intelligent Systems (CCIS)* (pp. 1-6). IEEE.

17. Falade, Polra Victor, and Grace Bunmi Ogundele. "Vulnerability analysis of digital banks' mobile applications." *arXiv preprint arXiv:2302.07586* (2023).

18. Farhang, S., Laszka, A., & Grossklags, J. (2018, February). An economic study of the effect of android platform fragmentation on security updates. In *International Conference on Financial Cryptography and Data Security* (pp. 119-137). Berlin, Heidelberg: Springer Berlin Heidelberg.

19. Ferrari, L., Pagano, F., Verderame, L., Romdhana, A., Caputo, D., & Merlo, A. (2025). The OWApp Benchmark: an OWASP-compliant Vulnerable Android App Dataset. *Authorea Preprints*.

20. Garg, S., & Baliyan, N. (2021). Comparative analysis of Android and iOS from security viewpoint. *Computer Science Review*, *40*, 100372.

21.. Gupta, B. B., Yamaguchi, S., & Agrawal, D. P. (2018). Advances in security and privacy of multimedia big data in mobile and cloud computing. *Multimedia Tools and Applications*, *77*, 9203-9208.

22. Haq, I. U., & Khan, T. A. (2021). Penetration frameworks and development issues in secure mobile application development: A systematic literature review. *IEEE Access*, *9*, 87806-87825.

23. Karo-Karo, G. F. M., Windarta, S., & Hikmah, I. R. (2024, November). Evaluating Compliance of the XYZ Ministry's Android Messaging Applications with OWASP MASVS: A Comprehensive Case Study. In *2024 IEEE 2nd International Conference on Electrical Engineering, Computer and Information Technology (ICEECIT)* (pp. 35-40). IEEE.

24. Kalauner, P. G. (2023). *Analysis and bypass of android application anti-reverse engineering mechanisms* (Doctoral dissertation, Technische Universität Wien).

25. Katoch, S., & Garg, V. (2023, March). Security analysis on android application through penetration testing using reverse engineering. In *2023 3rd International Conference on Smart Data Intelligence (ICSMDI)* (pp. 216-222). IEEE.

26. KAUR, G., & DHAWAN, A. (2024). *Laws of electronic evidence and digital forensics*. PHI Learning Pvt. Ltd.

27. Kemenuh, I. B. A. S., Huwae, R. B., & Jatmika, A. H. Security Analysis of the Lombok Tourism Android Application Using Penetration Testing (Pentesting) Methods Based on the OWASP Mobile Top 10-2024 Framework.

28. Khan, S., Yusuf, A., Haider, M., Thirunavukkarasu, K., Nand, P., & Rahmani, M. K. I. (2022, June). A review of android and ios operating system security. In *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)* (pp. 67-72). IEEE.

29. Mahboubi, A. (2018). *Security of critical information infrastructures exposed to mobile personal devices* (Doctoral dissertation, Queensland University of Technology).

30. Marguerite, D., & Patrick, M. (2022). Secure Mobile DevOps: Integrating Security from Code to Deployment in Mobile CI/CD Pipelines. *International Journal of Trend in Scientific Research and Development*, *6*(2), 1613-1619.

31. Mishra, A. (2022). *Mobile app reverse engineering*. Packt Publishing Limited.

32. Mutti, S., Fratantonio, Y., Bianchi, A., Invernizzi, L., Corbetta, J., Kirat, D., ... & Vigna, G. (2015, December). Baredroid: Large-scale analysis of android apps on real devices. In *Proceedings of the 31st Annual Computer Security Applications Conference* (pp. 71-80).

33. Nguyen-Vu, L., Chau, N. T., Kang, S., & Jung, S. (2017). Android rooting: An arms race between evasion and detection. *Security and Communication Networks*, *2017*(1), 4121765.

34. Noman, S. A., & Noman, H. A. (2017) An Empirical Study of Pentesting IOS 9 Applications. International Journal of Electrical & Computer Sciences IJECS-IJENS Vol:16 No:02

35. Nutalapati, V. (2023). Automated Security Testing for Mobile Apps: Tools, Techniques, and Best Practices. *International Research Journal of Engineering & Applied Sciences (IRJEAS)*, *11*(1), 26-31.

36. O'Dea, S. (2020). Market share of mobile operating systems worldwide 2012–2019. *Statista [Online]*.

37. ÖZGÜR, B., DOĞRU, İ. A., & Alkan, M. (2021, December). A Suggested Model for Mobile Application Penetration Test Framework. In *2021 International Conference on Information Security and Cryptology (ISCTURKEY)* (pp. 18-21). IEEE.

38. Parveen, M., & Shaik, M. A. (2023, August). Review on Penetration Testing Techniques in

39. Cyber security. In *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)* (pp. 1265-1270). IEEE.

40. Philip, J., & Raju, M. (2019). A formal overview of application sandbox in android and iOS with the Need to secure sandbox against increasing number of malware attack. *Indian Journal of Computer Science*, *4*(3), 32-40.

41. Priambodo, D. F., Ajie, G. S., Rahman, H. A., Nugraha, A. C. F., Rachmawati, A., & Avianti, M. R. (2022, November). Mobile health application security assesment based on owasp top 10 mobile vulnerabilities. In *2022 international conference on information technology systems and innovation (ICITSI)* (pp. 25-29). IEEE.

42. Ranganath, V. P., & Mitra, J. (2020). Are free android app security analysis tools effective in detecting known vulnerabilities? *Empirical Software Engineering*, *25*(1), 178-219.

43. Relan, K. (2016). *.: A Definitive Guide to IOS Security*. Apress.

44. Roshanaei, M. (2024). Enhancing mobile security through comprehensive penetration testing. *Journal of Information Security*, *15*(2), 63-86.

45. Saifulhakim, M., & Fajar, A. N. (2024). Security Assessment for Digital Wallet Payment Partner Applications using the OWASP Method: A Case Study in Indonesia. *Journal of System and Management Sciences*, *14*(3), 337-357.

46. Shi, P., Qin, F., Cheng, R., & Zhu, K. (2019, July). The penetration testing framework for largescale network based on network fingerprint. In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)* (pp. 378-381). IEEE.

47. Shishkova T (2021) IT threat evolution in Q3 2021. Mobile statistics. https://securelist.com/itthreat-evolution-in-q3-2021-mobile-statistics/105020/

48. Tanque, M. (2018). Security for Hybrid Mobile Development: Challenges and Opportunities. *Mobile Commerce: Concepts, Methodologies, Tools, and Applications*, 625-667.

49. Terneva, Z., & Dimitrova, S. (2023, September). Penetration testing of devices. In *2023 International Scientific Conference on Computer Science (COMSCI)* (pp. 1-4). IEEE.

50. Thomas, G., Burmeister, O., & Low, G. (2019). The Importance of Ethical Conduct by Penetration Testers in the Age of Breach Disclosure Laws. *Australasian Journal of Information Systems*, *23*.

51. Verma, P., & Dixit, A. (2016). *Mobile Device Exploitation Cookbook*. Packt Publishing Ltd.

52. Wu, X., Kumar, S., & Pang, M. S. (2020). Tackling Android Fragmentation: Mobile Apps' Dilemma and the Platform's Strategies.

53. Yamin, M. M., & Katt, B. (2019, January). Mobile device management (MDM) technologies, issues and challenges. In *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy* (pp. 143-147).

54. Zhang, J., Yao, Y., Li, X., Xie, J., & Wu, G. (2017). An android vulnerability detection system. In *Network and System Security: 11th International Conference, NSS 2017, Helsinki, Finland, August 21–23, 2017, Proceedings 11* (pp. 169-183). Springer International Publishing.

55. Zwilling, M., Klien, G., Lesjak, D., Wiechetek, Ł., Cetin, F., & Basim, H. N. (2022). Cyber security awareness, knowledge and behavior: A comparative study. *Journal of Computer Information Systems*, *62*(1), 82-97.