e-ISSN 2549-7472
Edumatic: Jurnal Pendidikan Informatika
Volume 10, Issue 1, **80-89.**
https://e-journal.hamzanwadi.ac.id/index.php/edumatic
DOI: 10.29408/edumatic.v10i1.33285

Security Maturity Assessment of Indonesian Android Mobile Banking Apps using MobSF and OWASP

Rizal Aglal Faozi [1], Nuur Wachid Abdul Majid [1,*], Suprih Widodo [1]

[1] Universitas Pendidikan Indonesia, Indonesia
[*] Corresponding author: Nuur Wachid Abdul Majid, Universitas Pendidikan Indonesia, Indonesia
✉ nuurwachid@upi.edu

Received: 3 December 2025 | Revised: 1 January 2026 | Accepted: 1 March 2026 | Published: 12 March 2026

Abstract
The rapid expansion of mobile banking in emerging economies has increased exposure to client-side security risks, while MASVS-based security maturity benchmarking of conventional banking applications remains underrepresented in the literature. This study conducts a standard-based comparative security maturity assessment of two major Indonesian Android banking applications, BRImo and myBCA. APK files obtained from the Google Play Store were analysed using Static Application Security Testing with the Mobile Security Framework (MobSF) and evaluated against OWASP MASVS Level 2 and MASVS-R. MobSF scores were interpreted as relative indicators of security maturity based on severity-weighted findings across multiple domains. The results reveal a clear divergence in maturity levels. Although both applications demonstrate strong network-layer protection, BRImo exhibits structural weaknesses in storage, cryptography, platform interaction, and resilience domains, indicating fragmented defence-in-depth implementation. In contrast, myBCA shows more consistent cross-domain control integration. This study contributes an MASVS-based security maturity benchmarking approach and provides conceptual evidence that formal regulatory compliance may coexist with inconsistent client-side technical implementation. The findings offer analytically transferable insights for developers, security auditors, and regulators in rapidly digitalising financial ecosystems.

Keywords: defence-in-depth; mobile banking security; owasp masvs; security maturity; static analysis

To cite this article: Faozi, R. A., Majid, N. W. A., & Widodo, S. (2026). Security Maturity Assessment of Indonesian Android Mobile Banking Apps using MobSF and OWASP. *Edumatic: Jurnal Pendidikan Informatika*, *10*(1), 80–89. https://doi.org/10.29408/edumatic.v10i1.33285

INTRODUCTION

Smartphones have become integral to daily life, reshaping how people communicate, work, study, and socialise to the point of becoming a primary psychological and functional attachment (Nawaz et al., 2026). Smartphones have evolved into multifunctional computing devices that enhance productivity, facilitate instant communication, support learning, provide entertainment, and enable financial transactions, driving widespread adoption over time (Cinar & Kara, 2023; Precht et al., 2024; Sánchez-Fernández & Borda-Mas, 2023). As smartphone-based financial services expand, strong application security becomes critical to protecting sensitive personal and financial data (Naeem et al., 2022). Mobile banking applications now process high-value transactions and confidential user data, making client-side vulnerabilities a direct threat to data confidentiality and systemic financial security. In Indonesia, national

cybersecurity reports indicate that the financial services sector consistently ranks among the most targeted industries, with hundreds of millions of cyber incidents reported annually, prompting stricter regulatory requirements through PP No. 71/2019, POJK No. 11/POJK.03/2022, UUP2SK, and OJK Regulation No. 3/2024.

Recent threat trends indicate an increasing shift of mobile banking risks from user behaviour toward client-side application vulnerabilities. Common client-side weaknesses frequently reported in mobile banking applications include insecure SSL/TLS implementation that may enable man-in-the-middle attacks, poor code quality that increases the likelihood of exploitable flaws, improper default permissions that unnecessarily broaden the application attack surface, unencrypted local storage that risks sensitive data exposure, outdated software components that introduce known vulnerabilities, and the continued use of weak or deprecated cryptographic algorithms (Abiola, 2023; Ali, 2026; Archibong et al., 2024; Imam et al., 2024). These recurring weaknesses suggest that many mobile banking applications still face challenges in implementing security controls consistently across the application lifecycle.

Existing studies on mobile application security have primarily focused on vulnerability identification, malware behaviour, or configuration weaknesses within specific application contexts. Research in this domain ranges from detecting hidden sensitive operations and inter-app collusion to preventing rooting-based data leakage through Java function hooking (El-Zawawy & Katsikas, 2026; Soewito & Suwandaru, 2022). Recent advancements have also introduced specialized testing for authentication flaws in GUIs and concurrent abstract interpretation for JavaScript vulnerabilities in Android WebViews (Amalfitano et al., 2025; Yuan et al., 2026). Furthermore, the field has evolved towards automated and collaborative detection, utilizing deep learning for malware-to-vulnerability mapping, as well as blockchain-based federated neural networks for real-time vulnerability identification with explainable AI (XAI) (Garg & Baliyan, 2022; Senanayake et al., 2024). While these works provide valuable technical insights, they largely treat security findings as isolated technical issues rather than as indicators of overall security maturity. In particular, prior research has not systematically examined whether security controls are implemented consistently across MASVS domains in high-adoption conventional mobile banking applications. As a result, the relationship between regulatory compliance, organisational scale, and actual client-side security maturity remains insufficiently characterised in the literature.

Accordingly, this study adopts OWASP MASVS as a defence-in-depth framework to provide both analytical and explanatory evaluation of mobile banking security. Unlike prior MobSF-based studies, this research reframes automated outputs as relative indicators of security maturity defined by the consistency, coverage, and integration of security controls rather than mere vulnerability counts. The objectives of this study are twofold: (1) to evaluate the client-side security maturity of high-adoption Indonesian mobile banking applications across MASVS domains, and (2) to examine whether institutional scale and regulatory context are reflected in consistent technical security implementation. By positioning Indonesia as a representative emerging economy facing legacy complexity and rapid digitalisation, this study offers generalizable insights into the security maturity of large-scale mobile banking ecosystems.

METHOD

This study applies a standard-based security assessment and comparative security maturity evaluation to examine the client-side security posture of mobile banking applications in Indonesia. The objective is to assess the implementation quality of security controls and defence-in-depth mechanisms in high-adoption mobile banking applications, focusing on observable application-layer characteristics rather than user behaviour or organisational policy.

Two Android mobile banking applications, BRImo version 2.75.0 and myBCA version 1.32.0, were selected as case studies, with APK files obtained from the Google Play Store in December 2024. These applications represent major digital banking user bases in Indonesia and are therefore suitable for comparative security maturity assessment. The analysis employs Static Application Security Testing (SAST) to examine application packages without execution (Alsumayt et al., 2024), enabling detection of code-level and configuration-level issues such as insecure cryptographic implementation, weak local data storage, excessive permissions, and insecure network configurations, while excluding runtime logic, backend, authentication bypass through dynamic behaviour, and server-side vulnerabilities; thus, the analysis is limited to client-side aspects.

Security analysis was conducted using the Mobile Security Framework (MobSF), selected for its reproducible automated analysis, wide academic and industrial adoption, and explicit mapping to OWASP MASVS controls. MobSF-generated security scores are treated as relative indicators for cross-application comparison rather than absolute security measures. The evaluation framework is based on OWASP MASVS, used as both a security maturity and defence-in-depth model, with emphasis on MASVS Level 2 (Defence-in-Depth) and MASVS-R (Resilience) requirements appropriate for sensitive financial applications (Holguera et al., 2023).

The research workflow begins with APK acquisition and proceeds through isolated environment preparation, MobSF deployment, controlled APK collection, full static analysis, vulnerability extraction and classification, MASVS domain mapping using an official reference, and manual validation of critical findings to reduce false positives, followed by comparative analysis using MobSF security scores.

This study is limited to client-side APK analysis and excludes dynamic behaviour, backend APIs, and server infrastructure. Although the small sample size limits statistical generalisation, it supports deep technical evaluation of critical systems. All APKs were obtained from official sources and analysed strictly for academic purposes, with any critical findings to be disclosed responsibly to the respective providers.

RESULTS AND DISCUSSION

Results

This section presents the results of a comparative security maturity assessment of two Indonesian Android mobile banking applications, BRImo and myBCA, based on static analysis using MobSF and evaluated against OWASP MASVS Level 2 and MASVS-R. In this study, the security score generated by MobSF is interpreted as a relative indicator of security maturity rather than an absolute security rating. The score reflects the cumulative impact of severity-weighted findings across multiple security domains, enabling comparative analysis of defence-in-depth implementation. The results are presented progressively, beginning with an overall maturity indicator, followed by domain-based analysis to highlight structural weaknesses, cumulative risk exposure, and inter-domain security imbalance.

Table 1 presents the fundamental characteristics and overall security assessment results of the BRImo and myBCA applications. Both applications target Android SDK 34 and require a minimum SDK of 24, indicating comparable platform compatibility and development baselines. Despite these similarities, a clear disparity is observed in the security scores generated by MobSF. BRImo obtained a score of 36 out of 100, while myBCA achieved a score of 60 out of 100. This difference reflects relative security maturity rather than the sheer number of detected vulnerabilities. A lower score indicates the accumulation of high-severity weaknesses across multiple security domains, suggesting systemic exposure rather than isolated misconfigurations. Accordingly, a limited number of high-severity findings

disproportionately shape overall security maturity, as they represent violations of core MASVS controls rather than isolated implementation or hygiene issues.

Table 1. Fundamental application information and security assessment results

Parameter	BRImo	MyBCA
App Name	Brimo	myBCA
Package Name	id.co.bri.brimo	com.bca.mybca.omni.android
Version	2.75.0	1.32.0
Version Code	652757	168
Size	90.53 MB	51.69 MB
Minimum SDK	24	24
Target SDK	34	34
Max SDK	Not Declared	Not Declared
Main Activity	id.co.bri.brimo.ui.activities .SplashScreenActivity	.presentation.splashscreenrevamp .SplashScreenActivity
App Security Score	36/100	60/100
Security Grade	C (High Risk)	A (Low Risk)
Tracker	3/432	3/432

Although the analysis identified numerous findings categorised as Info or Warning, the overall security posture of both applications is primarily influenced by a limited number of High-severity findings. High-severity issues represent direct violations of fundamental security controls and contribute disproportionately to cumulative risk exposure. In contrast, Info and Warning findings mainly indicate hygiene or best-practice issues. As a result, a small number of critical findings, such as insecure local storage and hardcoded secrets, have a significantly greater impact on security maturity than a large number of minor findings.

Table 2. Findings on storage aspects

Findings	Severity	BRImo	myBCA
Sensitive Logging	Info	✓	✓
Clipboard Leakage	Info	✓	✓
External Storage Access	Warning	✓	✓
SharedPreferences World-Readable	High	✓	-
SharedPreferences World-Writable	High	✓	-
Temporary File Creation	Warning	✓	✓
Hardcoded Sensitive Information	Warning	✓	✓

Table 2 presents storage-related findings mapped to the MASVS Storage domain. Both applications exhibit low- to medium-severity issues, including sensitive logging, clipboard leakage, external storage access, and temporary file creation, indicating common storage hygiene weaknesses. However, BRImo demonstrates additional high-severity findings related to world-readable and world-writable SharedPreferences, which were not observed in myBCA. These findings indicate that sensitive application data may be accessible to other applications on the same device, violating the Android sandboxing principle. This condition represents non-compliance with MASVS storage requirements, such as MSTG-STORAGE-2, which mandates secure and private local data storage. Consequently, local data protection failure is identified

as a primary and structural security weakness in BRImo rather than a minor implementation error.

Table 3 summarises cryptographic-related findings mapped to the MASVS Cryptography domain. Both applications were found to use insecure random number generators, which weakens the entire cryptographic system regardless of the strength of other algorithms. Poor randomness undermines key generation, token creation, and cryptographic protocol integrity. Additionally, weak hashing algorithms, specifically SHA-1 and MD5, were detected in myBCA, representing a latent vulnerability that may become exploitable under certain attack scenarios. Hardcoded secrets were identified in both applications, with BRImo containing a substantially higher number than myBCA. The presence of hardcoded secrets is interpreted as a proxy indicator of Secure SDLC failure, reflecting weaknesses in secret management governance rather than isolated coding mistakes. The use of an encrypted Realm database in myBCA indicates a more systematic approach to cryptographic data protection.

Table 3. Findings on cryptography aspects

Findings	Severity	BRImo	myBCA
Insecure Random Number Generator	High	✓	✓
Weak hashing (SHA-1, MD5)	High	-	✓
Hardcoded Secrets	High	69 items	26 items
Encrypted Realm Database	Secure	-	✓

Taken together, the findings in the Storage and Cryptography domains indicate weaknesses in data-at-rest governance, where insecure local storage and improper secret management undermine fundamental defence-in-depth controls. In this context, hardcoded secrets function as a proxy indicator of secure SDLC governance failure rather than isolated coding errors.

Table 4 presents findings related to network communication security. Both BRImo and myBCA implement SSL certificate pinning, satisfying MASVS network security requirements such as MSTG-NETWORK-4 and indicating mature perimeter security controls. However, BRImo exhibits high-severity findings related to insecure WebView handling. This contrast highlights a scientific contradiction between strong network security and weak client-side security. The results confirm that perimeter security alone is insufficient to ensure overall application security and reveal an imbalance in security maturity across MASVS domains.

Table 4. Findings on network communication aspects

Findings	Severity	BRImo	myBCA
SSL Pinning	Secure	✓	✓
Insecure Webview Handling	High	✓	-

Table 5 summarises privacy-related findings. Both applications request 25 permissions, including 15 dangerous permissions. While permission usage does not directly indicate exploitable vulnerabilities, the number of dangerous permissions expands the application's attack surface. In the presence of other vulnerabilities, this expanded surface increases the potential impact radius of a successful exploit. Therefore, permission configuration should be interpreted as a risk amplification factor rather than an isolated security flaw.

Table 5. Findings on privacy aspects

Findings	BRImo	myBCA
Total Permissions	25	25
Dangerous Permissions	15	15

.Table 6 presents resilience-related findings mapped to MASVS-R. Both applications exhibit high-severity findings related to debugging configurations. BRImo was found to have remote WebView debugging enabled, while myBCA was identified as running with debug configuration enabled. Debugging features in production environments significantly reduce the cost of attack by enabling runtime inspection, manipulation, and reverse engineering. These findings indicate high operational risk exposure and insufficient hardening of production builds, increasing exploit feasibility even for less sophisticated attackers. These conditions significantly lower the technical barrier to exploitation, expanding the threat model to include low-skilled attackers and increasing real-world exploit feasibility.

Table 6. Findings on resilience aspects

Findings	Severity	BRImo	myBCA
Remote Webview Debugging Enabled	High	✓	-
Debug Configuration Enabled	High	-	✓

The combination of cryptographic weaknesses and insufficient resilience controls amplifies overall security risk, as insecure randomness and hardcoded secrets become significantly more exploitable in environments where debugging features remain enabled and runtime protections are weak. Under these conditions, cryptographic assets can be extracted or manipulated through reverse engineering, transforming latent cryptographic flaws into practical attack vectors that undermine defence-in-depth.

Platform interaction findings are presented in Table 7. BRImo demonstrates multiple warnings related to unsafe WebView configurations, including enabled file access and potential code execution. These findings increase exposure to local file access and script execution risks within the WebView component. In contrast, myBCA implements tapjacking protection mechanisms and does not exhibit unsafe WebView-related findings. This contrast reflects differences in security governance maturity, where myBCA applies defensive controls more systematically, while BRImo accumulates cross-domain configuration weaknesses.

Table 7. Findings on platform interaction aspects

Findings	Severity	BRImo	myBCA
Unsafe WebView – File Access Enabled	Warning	✓	-
Unsafe WebView – Code Execution	Warning	✓	-
Tapjacking Protection	Secure		✓

When considered together, the Network and Platform Interaction domains reveal a security maturity imbalance, where strong perimeter protections such as SSL pinning coexist with weak client-side execution controls. This pattern indicates fragmented defence-in-depth implementation, where secure communication does not extend to secure content rendering and interaction handling.

Table 8 presents code quality-related findings. IP address disclosure was detected in the myBCA application, while no similar findings were identified in BRImo. Although this finding is lower in severity compared to storage or cryptographic weaknesses, it still represents information exposure that could facilitate reconnaissance activities. As such, it contributes to cumulative information leakage risk.

Table 8. Findings on code quality aspects

Findings	Severity	BRImo	myBCA
IP Address Disclosure	Warning	-	✓

To provide a consolidated overview, Table 9 summarises relative security maturity across MASVS domains. This summary highlights domains in which each application demonstrates structural strength or weakness prior to detailed interpretation. Domain-level security maturity was derived by correlating MASVS compliance status, severity-weighted findings, and cumulative risk exposure within each domain, rather than the absolute number of detected issues. A domain was classified as Strong when no high-severity findings were detected and core MASVS Level 2 controls were satisfied. Moderate indicates partial compliance with MASVS requirements, where medium-risk or hygiene-related findings exist but no structural high-risk misconfigurations were observed. Weak denotes the presence of high-severity or structural findings that violate core MASVS controls and significantly increase attack feasibility. Therefore, the domain-level maturity classification reflects qualitative differences in control effectiveness and defence-in-depth consistency, rather than purely quantitative differences in vulnerability counts.

Table 9. MASVS domain-level security maturity summary

MASVS Domain	BRImo	myBCA
Storage	Weak	Moderate
Cryptography	Weak	Moderate
Network	Strong	Strong
Platform Interaction	Weak	Moderate
Resilience	Weak	Weak
Code Quality	Moderate	Moderate

Discussion

This study provides empirical evidence that challenges the widely held assumption that mobile banking applications developed by large and highly regulated banks are inherently secure. Despite operating within the same regulatory and financial environment, BRImo and myBCA demonstrate markedly different levels of security maturity. This divergence indicates that organisational scale and formal compliance do not automatically translate into robust technical security at the application level. Instead, the findings suggest that security maturity is primarily determined by how systematically security controls are embedded and enforced throughout the software development lifecycle.

This study advances the conceptual understanding of mobile banking security maturity by demonstrating that defence-in-depth failure is often domain-asymmetric rather than uniform. This phenomenon aligns with the Weakest Link Theory, which posits that a system's security is constrained by its least secure component (Marques et al., 2024). The findings reveal a recurring pattern in which strong perimeter controls coexist with weak client-side governance, particularly in local data protection, cryptographic asset management, and runtime hardening. This asymmetry suggests that security maturity should be interpreted as the consistency of control enforcement across domains rather than the presence of isolated strong safeguards. Consequently, security maturity emerges as an organisational and process-level property shaped by governance and SDLC enforcement.

Although both applications demonstrate mature network-layer protection through SSL pinning, this strength is not consistently reflected across other security domains. This observation contradicts the Principle of Complete Mediation, which requires every access to be consistently validated (Ebad, 2022). Critical weaknesses were identified in local data protection and application resilience, indicating fragmented implementation of defence-in-depth rather than a security-by-design approach. Defence-in-depth requires coordinated protection across data at rest, data in transit, platform interaction, and runtime resilience. The coexistence of strong perimeter security with weak client-side controls confirms that network-

layer security alone is insufficient to ensure comprehensive application security, a principle explicitly emphasised in modern mobile security standards such as OWASP MASVS.

The most significant scientific contribution of this study lies in identifying local data protection failure as a primary security weakness, particularly in BRImo. The presence of world-readable and world-writable SharedPreferences represents a direct violation of the Android application sandboxing principle and MASVS Storage requirements. This issue can be analyzed through the lens of the "Secure by Default" principle, which dictates that the defaults should be safe and secure (Ruohonen, 2025). This issue is structural in nature, as it affects how sensitive data is persistently stored rather than resulting from isolated coding errors. These findings are consistent with prior empirical studies showing that insecure data storage remains a dominant weakness in mobile financial applications, affecting up to 82% of evaluated apps and indicating persistent challenges in secure data handling practices (Archibong et al., 2024).

The widespread presence of hardcoded secrets, particularly the substantially higher number detected in BRImo, should be interpreted as an indicator of Secure SDLC breakdown rather than isolated implementation mistakes. Embedding API keys, cryptographic keys, or tokens directly into client-side code reflects insufficient governance over secret lifecycle management and inadequate pre-release security auditing. This reflects violations of Least Privilege and Separation of Duties, where sensitive credentials remain embedded in client code. Such hardcoded credentials frequently lead to service exposure and backend compromise, marking the issue as a systemic development process failure rather than a simple bug. These vulnerabilities are further exacerbated by systemic risk amplifiers in the form of cryptographic weaknesses; specifically, the use of insecure random number generators undermines the entropy guarantees required for secure key generation and token creation, while weak hashing algorithms reduce long-term cryptographic assurance, even when immediate exploitation is not observed (English et al., 2024).

Resilience findings expand the threat model. In accordance with Shannon's Maxim, a secure system must be designed under the assumption that 'the enemy knows the system.' The presence of active debugging in production, however, effectively hands the adversary a roadmap to the application's inner workings. When coupled with hardcoded secrets, this eliminates the cost of discovery, transforming what should be a complex reverse-engineering task into a trivial extraction process. This aligns with findings by Park et al. (2024), who confirm that the absence of runtime protections materially increases tampering feasibility. Collectively, weaknesses in storage, cryptography, and resilience form a cross-domain pattern of fragmented security governance that amplifies exploitability.

A comparative examination of BRImo and myBCA reveals differences that extend beyond individual vulnerabilities. myBCA demonstrates a more systematic approach to security governance, as reflected in encrypted local databases and tapjacking protection, whereas BRImo exhibits an accumulation of weaknesses across multiple security domains. This contrast suggests differences in security culture, pre-release auditing practices, and organisational commitment to secure software development, indicating that divergence in security scores reflects governance maturity rather than isolated technical deficiencies.

The findings also reveal a disconnect between regulatory compliance and technical implementation within the Indonesian banking context. While financial institutions may adhere to regulatory and governance requirements, such compliance does not necessarily ensure consistent enforcement of technical security controls at the application level. Regulatory frameworks often prioritise policy and organisational governance, whereas application security requires continuous technical validation. Overall, this discussion demonstrates that security maturity in mobile banking applications is shaped more by governance consistency and SDLC enforcement than by organisational scale or regulatory status, offering a conceptual explanation

for why compliance-oriented approaches may fail to deliver robust application-layer protection.

CONCLUSION

This study evaluates the security maturity of Indonesian Android mobile banking applications using MobSF and OWASP MASVS Level 2 and MASVS-R. The results demonstrate a clear divergence in security maturity between BRImo and myBCA, indicating that strong network-layer protection does not necessarily translate into robust data-at-rest security, cryptographic governance, or application resilience. Key findings such as insecure local storage, hardcoded secrets, cryptographic weaknesses, and debugging configurations in production builds reveal fragmented defence-in-depth implementation and weaknesses in SDLC enforcement. The main scientific contributions of this study are threefold: providing an empirical MASVS-based security maturity benchmarking for conventional mobile banking applications, introducing the concept of domain-asymmetric security maturity failure rooted in governance and SDLC enforcement, and extending global mobile application security findings to the context of large banks in emerging economies. The findings offer analytically transferable insights for developers, security auditors, and regulators, highlighting the need for severity-weighted, cross-domain security evaluation and continuous technical verification beyond regulatory compliance. Future work should integrate Dynamic Application Security Testing (DAST) and backend API assessment to enable comprehensive end-to-end security evaluation.

REFERENCES

Abiola, O. B. (2023). Exploring Mobile Banking App Security from User's Perspectives. *International Journal for Information Security Research*, *13*(1), 1077–1084. https://doi.org/10.20533/ijisr.2042.4639.2023.0122

Ali, A. O. M. M. (2026). The Security and Privacy of Mobile Banking in Qatar: A Case Highlighting Current Challenges and Future Recommendations. *American Journal of Multidisciplinary Research and Innovation*, *5*(1), 51–59. https://doi.org/10.54536/ajmri.v5i1.3162

Alsumayt, A., Elbeh, H., Elkawkagy, M., Alfawaer, Z., Alghamedy, F. H., Alshammari, M., Aljameel, S. S., Albassam, S., AlGhareeb, S., & Alamoudi, K. (2024). A Study of Android Security Vulnerabilities and Their Future Prospects. *HighTech and Innovation Journal*, *5*(3), 854–869. https://doi.org/10.28991/HIJ-2024-05-03-020

Amalfitano, D., Júnior, M., Fasolino, A. R., & Delamaro, M. (2025). A GUI-based Metamorphic Testing Technique for Detecting Authentication Vulnerabilities in Android Mobile Apps. *Journal of Systems and Software*, *224*, 112364. https://doi.org/10.1016/j.jss.2025.112364

Archibong, E. E., Stephen, B. U.-A., & Asuquo, P. (2024). Analysis of Cybersecurity Vulnerabilities in Mobile Payment Applications. *Archives of Advanced Engineering Science*, 1–12. https://doi.org/10.47852/bonviewaaes42022595

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), 20269–20281. https://doi.org/10.1007/s11042-023-14400-6

Ebad, S. A. (2022). Exploring How to Apply Secure Software Design Principles. *IEEE Access*, *10*, 128983–128993. https://doi.org/10.1109/ACCESS.2022.3227434

El-Zawawy, M. A., & Katsikas, S. (2026). Detecting Hidden Sensitive Operation vulnerabilities and their collusion inter-app attacks in Android. *Computers and Electrical Engineering*, *129*, 110794. https://doi.org/10.1016/j.compeleceng.2025.110794

English, K. V., Bennett, N., Thorn, S., Butler, K. R., Enck, W., & Traynor, P. (2024, June).

Examining cryptography and randomness failures in open-source cellular cores. *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy* (pp. 43-54). ACM. https://doi.org/10.1145/3626232.3653259

Garg, S., & Baliyan, N. (2022). M2VMapper: Malware-to-Vulnerability mapping for Android using text processing. *Expert Systems with Applications*, *191*, 116360. https://doi.org/10.1016/j.eswa.2021.116360

Holguera, C., Mueller, B., Schleier, S., & Willemsen, J. (2023). *OWASP Mobile Application Security Verification Standard* (Version 1.5). OWASP Foundation.

Imam, A. Y., Usman, H. I., & Abba, A. (2024). Security analysis and evaluation of mobile banking applications in Nigeria. *International Journal of Informatics and Communication Technology*, *13*(3), 354–361. https://doi.org/10.11591/ijict.v13i3.pp354-361

Marques, J. M. E., Mžourek, M., Papuga, J., Růžička, M., & Benasciutti, D. (2024). A probabilistic stress-life model supported by weakest link principle and highly-stressed volume/surface area concepts. *International Journal of Fatigue*, *178*, 108006. https://doi.org/10.1016/j.ijfatigue.2023.108006

Naeem, M., Ozuem, W., & Ward, P. (2022). Understanding the accessibility of retail mobile banking during the COVID-19 pandemic. *International Journal of Retail & Distribution Management*, *50*(7), 860-879. https://doi.org/10.1108/IJRDM-02-2021-0064

Nawaz, S., Linden, T., Mitchell, M., & Bhowmik, J. (2026). Examining effectual, ineffectual and problematic smartphone use: A qualitative exploration of dependence and management behaviours. *Social Sciences & Humanities Open*, *13*, 102569. https://doi.org/10.1016/j.ssaho.2026.102569

Park, Y., Choi, S., Choi, U. Y., Jin, H., Nor, N. H. M., & Park, Y. (2024). A practical approach for finding anti-debugging routines in the Arm-Linux using hardware tracing. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-65374-w

Precht, L.-M., Mertens, F., Brickau, D. S., Kramm, R. J., Margraf, J., Stirnberg, J., & Brailovskaia, J. (2024). Engaging in physical activity instead of (over)using the smartphone: An experimental investigation of lifestyle interventions to prevent problematic smartphone use and to promote mental health. *Journal of Public Health*, *32*(4), 589–607. https://doi.org/10.1007/s10389-023-01832-5

Ruohonen, J. (2025). SoK: The design paradigm of safe and secure defaults. *Journal of Information Security and Applications*, *90*, 103989. https://doi.org/10.1016/j.jisa.2025.103989

Sánchez-Fernández, M., & Borda-Mas, M. (2023). Problematic smartphone use and specific problematic Internet uses among university students and associated predictive factors: A systematic review. *Education and Information Technologies*, *28*(6), 7111–7204. https://doi.org/10.1007/s10639-022-11437-2

Senanayake, J., Kalutarage, H., Petrovski, A., Piras, L., & Al-Kadri, M. O. (2024). Defendroid: Real-time Android code vulnerability detection via blockchain federated neural network with XAI. *Journal of Information Security and Applications*, *82*, 103741. https://doi.org/10.1016/j.jisa.2024.103741

Soewito, B., & Suwandaru, A. (2022). Android sensitive data leakage prevention with rooting detection using Java function hooking. *Journal of King Saud University - Computer and Information Sciences*, *34*(5), 1950–1957. https://doi.org/10.1016/j.jksuci.2020.07.006

Yuan, Z., Yang, Z., Tan, J., & Zhang, H. (2026). WebViewJSdetect: Javascript vulnerability detection in android webview via coverage-guided thread-adaptive concurrent abstract interpretation. *Computer Networks*, *275*, 111908. https://doi.org/10.1016/j.comnet.2025.111908