Universal Library of Innovative Research and Studies Research Article

ISSN: 3065-0003 | Volume 1, Issue 2

Open Access | PP: 111-117

DOI: https://doi.org/10.70315/uloap.ulirs.2024.0102014

Ensuring the Security of the Mobile Application Supply Chain in the Context of the Evolution of Modern Mobile Threat Models

Aleksandr Pinaev

CEO and Founder of Swordfish Security and Mobix, Dubai, United Arab Emirates.

Abstract

This article analyses security threats specific to the mobile application supply chain during 2021–2024. The aim of the study is to systematise the contemporary attack landscape targeting mobile application supply chains and to propose a multi-level conceptual security model spanning all stages of the application lifecycle. The study employs a systematic literature review, a comparative analysis of industry reports, and case-study methodology. The study results show that third-party partners account for 15% of security breaches, with 63% of organizations experiencing significant losses from mobile-related incidents. Based on analysis of OWASP Mobile Top 10 (2024), the Keenadu and CocoaPods attack cases, and malicious SDK campaigns, the author proposes the Mobile Supply Chain Security Architecture (MSCSA), a five-level control model encompassing development, build, distribution, runtime, and governance. The findings are relevant to cybersecurity researchers, mobile application developers, IT risk management professionals, and regulators developing software security requirements.

Keywords: *Supply Chain Security, Mobile Applications, Mobile Threat Models, OWASP Mobile Top 10, Third-Party SDKs, Firmware Attacks, SBOM, DevSecOps, RASP, EU Cyber Resilience Act.*

INTRODUCTION

Mobile apps have become the primary interface of the digital economy, with smartphones projected to be used by more than 7.5 billion people by 2025 and annual app downloads exceeding 250 billion [1]. In parallel, the architecture of mobile software creation has grown substantially more complex: the average enterprise application incorporates between 30 and 100 third-party SDKs, open-source components, and dependencies sourced from diverse repositories [2]. This complexity creates a fundamentally new attack surface, the mobile application supply chain encompassing every stage from source-code authoring to application execution on the end-user device.

According to the Verizon 2024 Mobile Security Index, more than half of organizations experiencing downtime following a mobile security incident reported serious impacts, up from 47% in 2023 [7]. The Verizon 2024 DBIR found third-party involvement in 30% of breaches analyzed, up from approximately 15% the previous year [3]. Zimperium's 2024 Global Mobile Threat Landscape Report [4] found that 23.5% of enterprise devices contain apps installed outside of official stores, and 23% of enterprise apps interact with servers in high-risk jurisdictions. The updated OWASP Mobile Top 10 list [5] for the first time identified inadequate supply chain security (M2) as a separate critical threat category, reflecting the consensus of the professional community regarding the systemic nature of the problem.

Despite an actively evolving regulatory framework, the EU Cyber Resilience Act (2024) with mandatory SBOM requirements, the NIS2 Directive, and U.S. NIST SSDF standards [6], a significant gap persists in the academic literature. Most published research focuses either on the security of web-oriented software supply chains [7] or on isolated components of the mobile ecosystem, such as malware analysis [8] or SDK privacy [9]. A comprehensive, multi-level model addressing the specifics of mobile supply chains from development to on-device execution has not been systematically developed in the academic literature to date.

The objective of this study is to propose a conceptual Mobile Supply Chain Security Architecture (MSCSA) that provides a structured approach to risk reduction across all levels of the mobile application supply chain, grounded in a systematic analysis of the current threat landscape, real-world cases, and existing standards.

The scientific novelty of the work lies in the development of an original five-level conceptual MSCSA model that integrates threat vectors, controls, and tooling specific to the mobile lifecycle, distinguishing it from existing approaches oriented primarily toward server-side or generic software supply chains.

The author's hypothesis holds that fragmentation of countermeasures and the absence of an end-to-end security mechanism spanning all five levels of the mobile supply chain is the principal cause of sustained incident growth, and that addressing this through an architectural approach will yield qualitatively superior protection compared with reactive point solutions.

MATERIALS AND METHODS

The study is based on a comprehensive methodology combining systematic literature review, comparative analysis, case-study research, and content analysis of technical documentation.

The source base is organised into three categories. The first comprises peer-reviewed academic publications from IEEE Xplore, ACM Digital Library, and Springer Link covering 2020–2024. Selection used the keywords "mobile supply chain security", "third-party SDK vulnerability", "mobile threat model", "software bill of materials mobile", and "Android firmware security". From an initial pool of 187 publications, 14 academic sources were retained after applying inclusion criteria [2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

The second category consists of industry analytical reports from leading organisations: Verizon DBIR 2024 [3], Zimperium Global Mobile Threat Report 2024 [4], Guardsquare/Enterprise Strategy Group 2024 [20], and normative documents OWASP Mobile Top 10 2024 [5] and NIST SSDF [6], EU CRA, and SLSA (Supply-Chain Levels for Software Artifacts).

The third category comprises primary technical documents: OWASP Mobile Application Security Verification Standard (MASVS) [5], CycloneDX SBOM specifications, Sigstore documentation, and incident reports from Kaspersky, GitGuardian, and NowSecure.

The systematic literature review followed an adapted PRISMA protocol: database search, deduplication, title-and-abstract filtering, and full-text reading of selected works. The OWASP Mobile Top 10 (2024) [5] classification scheme was used to structure the threat landscape as the most current taxonomic instrument available.

Comparative analysis involved juxtaposing Verizon DBIR 2023–2024 [3] and Zimperium Reports 2022–2024 [4] time-series data to identify incident-growth trends. The analysis of five real incidents (Keenadu, CocoaPods, GitHub secrets, malicious advertising SDKs, attacks on banking applications) was carried out according to a single scheme: attack vector, scale of damage, tactics, and recording of the means used [3, 4, 20].

For the MSCSA conceptual model, structural decomposition was used: the supply chain was broken into five functional levels analogous to the SLSA model [6], and current attack vectors were mapped to corresponding controls at each level. Content analysis of technical documentation verified the compatibility of proposed controls with existing standards and tools.

RESULTS AND DISCUSSION

Statistics for 2021–2024 and forecasts for 2025 demonstrate a steady and accelerating growth in mobile app supply chain security threats. Key indicators captured in industry research are presented in Figure 1.

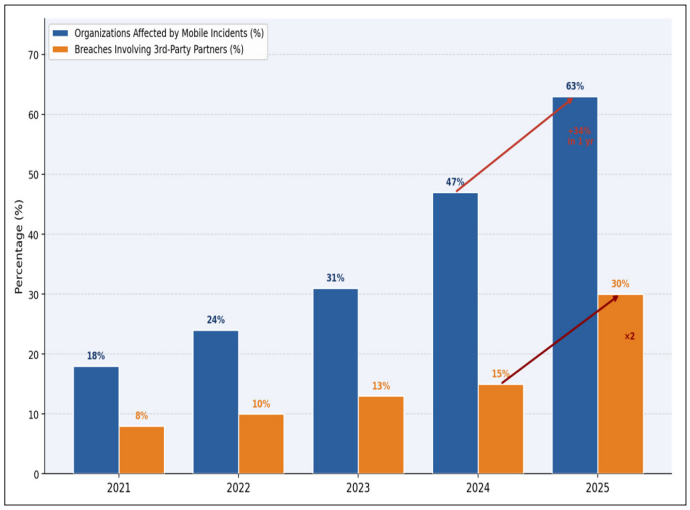

Figure 1. Growth of Mobile Supply Chain Security Incidents (compiled by the author based on [3]).

The data reflect a qualitative shift in the threat structure: if in 2021 mobile supply chain attacks were viewed as a peripheral risk, by 2024 they had become one of the principal vectors of corporate compromise. Particularly striking is the doubling of the third-party-related breach share from 15% to 30% within a single year [3]. This growth correlates with the entry of alternative app stores into the market (as a result of regulatory pressure on Apple and Google in the EU) and the widespread adoption of cross-platform frameworks that substantially deepens developer dependence on third-party components [7].

According to the Zimperium Global Mobile Threat Report 2024 [4], approximately 25% of enterprise devices contain sideloaded applications, while 29% of mobile devices encountered at least one malware attack during 2024, with banking trojans showing the fastest growth at 29% year-over-year. The "confidence gap" is equally telling: 93% of organisations are confident in the adequacy of their mobile defences, yet 62% of them experienced at least one mobile application security incident during the year [20].

The most significant taxonomic event was the 2024 update to OWASP Mobile Top 10, the first revision since 2016 [5]. The introduction of M2 (Inadequate Supply Chain Security) as a standalone threat class signals professional-community maturity. Table 1 presents a mapping of the top-5 OWASP Mobile Top 10 (2024) categories to specific supply chain attack vectors.

Table 1. OWASP Mobile Top 10 (2024) Threat Classification in the Context of Supply Chain Attacks (compiled by the author based on [3-5]).

	Threat Category (OWASP Mobile Top 10, 2024)	Supply Chain Attack Vector	Representative Exploit Example
1	Improper Credential Usage	Hardcoded API keys in third-party SDKs; secrets leakage in CI/CD pipelines	23M+ secrets leaked on GitHub, including mobile SDK keys
2	Inadequate Supply Chain Security	Malicious dependencies in npm/PyPI/CocoaPods; APK signing mechanism compromise	Keenadu: firmware-level malware via Android Zygote process
3	Insecure Authentication / Authorization	OAuth token bypass via malicious SDK; authorization response tampering	Compromised ad network SDK intercepts session tokens
4	Insufficient Input/Output Validation	Malicious data injection via analytics libraries; clipboard manipulation	Ad aggregator SDK embeds JavaScript injections into WebView components
5	Insecure Communication	SDKs transmitting data to sanctioned-country servers; missing TLS pinning	23% of enterprise apps communicate with high-risk servers

Five key incidents were analysed in depth to illuminate practical attack mechanisms. Their comparative characteristics are presented in Table 2.

Table 2. Comparative Analysis of Major Mobile Application Supply Chain Incidents (compiled by the author based on [3, 4, 10, 19]).

Incident	Year / Source	Attack Vector	Impact / Scale	Practical Lesson
Keenadu / Android Firmware	2024, Kaspersky	OEM firmware stage compromise; malware integrated via Zygote process	Dozens of manufacturers; malware copies into every app on device	Firmware integrity verification; independent OEM supply chain audit
Malicious CocoaPods Packages	2024, Guardsquare	Takeover of abandoned pods; malicious code injected into thousands of iOS apps	~3M apps potentially based on affected pods	Regular dependency auditing; dependency hash locking
GitHub Secrets Leakage	2024, GitGuardian	API key disclosure in mobile app source code	23M+ new secrets in one year; includes mobile SDK keys	CI/CD secret scanning; vault-based key management
Malicious Ad SDK Campaign	2024, Zimperium	Ad SDKs with hardcoded C2 servers in high-risk jurisdictions	23% of enterprise apps; data exfiltration and MDM bypass	SDK traffic behavioral analysis; security-vetted vendor selection
Banking App Supply Chain Attack	2024, Verizon DBIR	Compromised third-party component injects overlay malware	Credential theft across 12 banking apps; ~$6.99M avg. loss	Real-time behavioral monitoring; RASP for financial apps

The Keenadu malware incident represents, in the author's assessment, the most serious supply chain threat type: compromise occurred at the Android firmware level before the device reached the end user [3, 4]. The mechanism of exploiting the Zygote process, the system process that spawns all Android user applications, means every installed application is automatically infected without any user action. The previously documented Triada trojan employed an analogous vector [8]. A particular challenge is that enterprise security measures oriented at the application layer (MAM/MDM) are fundamentally incapable of detecting or neutralising threats of this class.

The open-source dependency manager attack vector is equally alarming. Vulnerabilities discovered in the CocoaPods ecosystem in 2024 [7] demonstrated that seizure of abandoned or deprecated pods allows an attacker to inject malicious code into potentially millions of iOS applications [7]. Attacks on npm, PyPI, and Maven similarly revealed the systemic nature of the threat across the entire open-source mobile development ecosystem [9]. GitGuardian data show that 23+ million new secrets were exposed on GitHub in 2024, including mobile SDK keys [10], directly validating the M1 position in the updated OWASP standard.

Authorial synthesis of the cases yields the following pattern: in the majority of recorded incidents, the vulnerability was exploited not through an error by the direct application

developer, but via a trusted component, an SDK, dependency, build tool, or device firmware. This means that traditional security models focused on the application's own code are fundamentally insufficient to counter supply chain threats [16, 18].

The proposed taxonomy differs from existing classifications oriented toward server-side or generic software supply chains in several key respects. First, it incorporates the mobile-platform-specific OEM firmware and pre-installed application layer, which lies beyond the control of both application developers and enterprise security teams. Second, the taxonomy accounts for the fragmentation of the mobile ecosystem: the same SDK can behave differently depending on OS version, device manufacturer, and app store configuration [11]. Third, it includes the direct-to-consumer vector, whose relevance has grown substantially following the opening of alternative app stores in the EU in 2024 [7].

Literature analysis [11, 12, 13] confirms a fundamental barrier to mobile supply chain security: information asymmetry. On server and desktop systems, security teams deploy monitoring agents, network probes, and inventory tools. On mobile devices, the security models of iOS and Android intentionally restrict visibility depth: applications are sandboxed, firmware and chipset solutions are opaque, and pre-installed OEM applications cannot be removed or replaced by enterprise MDM tools.

Based on the threat landscape analysis, case studies, and existing standards, the author proposes a conceptual Mobile Supply Chain Security Architecture (MSCSA), a five-level embedded security model for the mobile supply chain. The MSCSA model is built on the principle of "security as a property" embedded in every lifecycle stage, as opposed to a reactive "security as a bolt-on" approach. Each of the five levels has its own threat vectors, controls, tool support, and links to regulatory requirements. The detailed characteristics of each level are presented in Table 3.

Table 3. Detailed Characteristics of MSCSA Conceptual Model Levels (compiled by the author based on [5, 6, 14, 15, 20]).

Model Level	Components	Key Controls	Tools & Standards
L1: Development	Source code, dependencies, third-party SDKs	SAST scanning, SCA dependency checks, dependency hash policies	OWASP MASTG, GitHub Advanced Security, Snyk
L2: Build	CI/CD pipelines, build scripts, build artifacts	Artifact signing, hermetic build environment, provenance attestation	SLSA Level 2/3, Sigstore, Jenkins Security Plugin
L3: Distribution	App stores, CDN, mirrors	App signature verification, digital certificate control, SBOM	Google Play Integrity API, Apple Notarization, CycloneDX SBOM
L4: Runtime	Device environment, OEM firmware, SDKs	RASP, app integrity detection, behavioral monitoring	Guardsquare DexGuard, Approov, NowSecure Platform
L5: Governance	Policies, audit, compliance	Third-party component registry, responsible disclosure, vendor vetting	EU CRA, NIST SSDF, ISO/IEC 27034

A key innovation of the MSCSA model is the introduction of a unifying horizontal layer, the SBOM and Threat Intelligence Layer, that cuts across all five vertical levels. The practice of Software Bill of Materials (SBOM) generation in CycloneDX or SPDX format, mandatory under EU CRA from 2024 [6], enables an inventory registry of all application components: from primary to third-level transitive dependencies. Combined with continuous vulnerability monitoring (integration with NVD, OSV, and GitHub Advisory Database), SBOM transforms from a static document into a dynamic risk-management tool.

This approach has concrete measurable value: NowSecure research [17] demonstrates that organisations applying SBOM in combination with binary application analysis detect hidden SDKs, outdated cryptographic implementations, and hardcoded secrets that elude static source-code analysis. This is because a significant portion of SDKs is distributed in binary format without accessible source code, rendering traditional SAST fundamentally insufficient [11].

A second original element of the model is the explicit delineation of the Governance level, tied to a third-party component registry, responsible disclosure procedures, and vendor qualification. Existing frameworks, including OWASP MASVS [5] and SLSA [6], describe technical controls in detail but give insufficient attention to the process and organisational dimensions of supply chain management. The proposed model addresses this gap by including procedures for regular SDK vendor auditing, mandatory disclosure of data-collection practices in documentation (analogous to App Store and Google Play Privacy Label requirements), and third-party incident response protocols.

The author's analysis of threat evolution across the five supply chain levels for 2021–2024 is visualised in Figure 2 as a radar chart. Threat severity scores on a 1–10 scale were obtained by synthesising data from OWASP Mobile Top 10 (2024) [5], Verizon DBIR 2024 [3], and Zimperium 2024 [4].

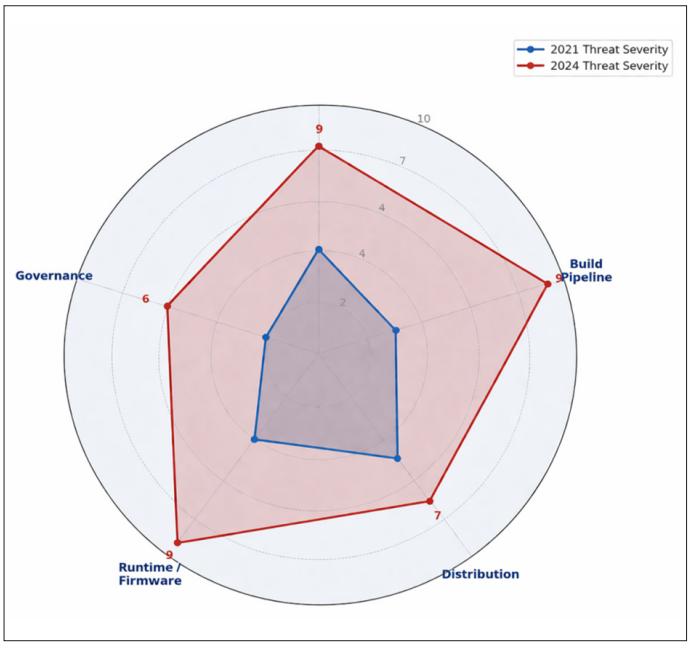

Figure 2. Mobile Supply Chain Threat Severity Evolution: 2021 vs. 2024 (compiled by the author based on [3-5]).

The diagram clearly shows that the most significant threat severity growth was recorded at the Build level (from 3 to 9) and Runtime level (from 4 to 9). Growth in build-level threats is driven by the industry's mass migration to cloud CI/CD pipelines and expanded use of cross-platform frameworks (Flutter, React Native, Kotlin Multiplatform), where a single attack can compromise applications on multiple platforms simultaneously. Growth in runtime-level threats reflects the evolution of firmware-level injection techniques (Keenadu, Triada) and the emergence of agentic AI tools capable of automating complex, multi-step attacks against mobile applications [7].

The relatively moderate growth in distribution-level threats (from 5 to 7) is explained by Apple and Google's substantial improvements to their store defences: Google Play Protect performs 125 billion app scans per day [3], while Apple has expanded its Notarization program. However, the opening of alternative app stores in the EU creates a new attack surface not covered by official platform protections [7], which implies likely further growth in distribution-level threats in 2026–2027.

At the Governance level, a positive shift is observed relative to other levels: the adoption of EU CRA, NIST SSDF, and mandatory SBOM requirements has created a formal regulatory framework that was absent in 2021. Nonetheless, in the author's assessment, practical implementation of these requirements lags considerably behind the pace of technical-level threat growth [6].

Realising a comprehensive approach to mobile supply chain security involves several structural barriers. The first and most fundamental is the information asymmetry between application developers and SDK providers: most commercial SDKs are distributed in binary format without public source code, depriving the developer of the ability to conduct a full security audit. Studies of Android application analysis show that the average application contains more than 60 third-party libraries, a significant proportion of which are included transitively and not explicitly declared [12].

The second barrier is Android ecosystem fragmentation: device manufacturers, carriers, and chipset vendors add their own software layers, creating unpredictable interactions with application components [11]. The inverse problem in the iOS ecosystem, while its closedness reduces fragmentation risks, the opacity of Secure Enclave components and device attestation mechanisms creates blind spots for enterprise security tools.

The third barrier is economic incentives distorted against security. The business model of most "free" SDKs is based on monetising user data via embedded advertising and analytics modules [9]. This creates a structural conflict of interest: the more granular the data collected by the SDK, the higher its commercial value to the provider, but also the greater the risk to users and organisations integrating it.

The fourth barrier is a shortage of automated threat-detection tools for mobile-specific CI/CD supply chain monitoring. Most existing SAST/DAST tools are optimised for web applications or server-side software and do not account for the specifics of mobile runtime: ART/Dalvik features in Android, Objective-C runtime in iOS, or cross-platform wrappers [12].

CONCLUSION

The study achieved its stated objective: based on a systematic analysis of the contemporary threat landscape (2021–2024), five high-profile real attack cases, and existing security standards, the author developed the Mobile Supply Chain Security Architecture (MSCSA), a conceptual framework providing a structured, multi-level approach to risk reduction across all stages of the mobile application lifecycle.

The study confirmed the authorial hypothesis: the most significant systemic factor driving incident growth is not the sophistication of individual attacks, but the fragmentation of countermeasures and the absence of an end-to-end security mechanism spanning all five supply chain levels from source code to on-device runtime. The SBOM-oriented component management, multi-level controls, and horizontal threat intelligence integration realised in MSCSA qualitatively outperform reactive point solutions in resilience against this attack class.

The key findings are as follows. First, the share of third-party-related breaches doubled within one year (from 15% to 30%, Verizon DBIR, 2024 [3]), signalling that supply chain threats have become a priority attack vector. Second, the introduction of OWASP Mobile Top 10 (2024) [5] category M2 (Inadequate Supply Chain Security) reflects professional

consensus on the systemic nature of the threat. Third, the most dangerous vector is OEM-firmware-level compromise (Keenadu/Triada class), against which most enterprise MDM/MAM controls are fundamentally ineffective. Fourth, the transition to mandatory SBOM requirements under EU CRA creates a normative basis for implementing the proposed MSCSA model in corporate practice.

The practical significance of the work is defined by the applicability of the MSCSA model as guidance for mobile application developers designing DevSecOps processes, for CISOs assessing and auditing third-party application supply chains, and for regulators developing industry mobile software security standards.

Directions for future research include: (1) empirical validation of the proposed MSCSA taxonomy through quantitative analysis of real CVE and incident data; (2) development of automated malicious-dependency detection methods specific to mobile CI/CD pipelines; (3) investigation of the regulatory effectiveness of SBOM requirements applied to the mobile ecosystem.

REFERENCES

1. Degenhard, J. (2024). Number of smartphone users worldwide 2014–2029 [Dataset]. Statista. Retrieved from: https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world (date accessed: August 20, 2024).
2. Pybus, J., & Coté, M. (2024). Super SDKs: Tracking personal data and platform monopolies in the mobile ecosystem. Big Data & Society, 11(1). https://doi.org/10.1177/20539517241231270.
3. Verizon Business. (2024). 2024 Data Breach Investigations Report (DBIR). Retrieved from: https://www.verizon.com/business/resources/reports/dbir.html (date accessed: August 22, 2024).
4. Zimperium. (2024). 2024 Global Mobile Threat Report. Retrieved from: https://lp.zimperium.com/hubfs/Reports/Global%20Mobile%20Threat%20Report%202024%20FINAL%20%281%29.pdf (date accessed: November 20, 2024).
5. OWASP Foundation. (2024). OWASP Mobile Top 10 (2024 edition). Retrieved from: https://owasp.org/www-project-mobile-top-10/ (date accessed: November 8, 2024).
6. Souppaya, M., Scarfone, K., & Dodson, D. (2022). Secure Software Development Framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities (NIST Special Publication 800-218). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-218.
7. Verizon Business. (2024). 2024 Mobile Security Index. Retrieved from: https://www.verizon.com/business/resources/T6bb/reports/2024-mobile-security-index.pdf (date accessed: September 12, 2024).
8. Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. IEEE Communications Surveys & Tutorials, 17(2), 998–1022. https://doi.org/10.1109/COMST.2014.2386139.
9. He, Y., Yang, X., Hu, B., & Wang, W. (2019). Dynamic privacy leakage analysis of Android third-party libraries. Journal of Information Security and Applications, 46, 259–270. https://doi.org/10.1016/j.jisa.2019.03.014.
10. GitGuardian. (2024). State of Secrets Sprawl 2024. Retrieved from: https://www.gitguardian.com/state-of-secrets-sprawl-report-2024 (date accessed: September 26, 2024).
11. JFrog. (2024). Software Supply Chain State of the Union 2024. Retrieved from: https://s201.q4cdn.com/814780939/files/doc_presentations/2024/2023-ssc-state-of-the-union_march2024.pdf (date accessed: October 4, 2024).
12. Yang, S., Bai, G., Lin, R., Guo, J., & Diao, W. (2024). Beyond the horizon: Exploring cross-market security discrepancies in parallel Android apps. In 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE) (pp. 558–569). https://doi.org/10.1109/ISSRE62328.2024.00059.
13. Meng, M. H., Yan, C., Hao, Y., Zhang, Q., Wang, Z., Wang, K., Teo, S. G., Bai, G., & Dong, J. S. (2024). A large-scale privacy assessment of Android third-party SDKs. arXiv. https://doi.org/10.48550/arXiv.2409.10411.
14. Yan, C., Meng, M. H., Xie, F., & Bai, G. (2024). Investigating documented privacy changes in Android OS. Proceedings of the ACM on Software Engineering, 1(FSE), 2701–2724. https://doi.org/10.1145/3660826.
15. Chen, Z. (2024). A comprehensive study of privacy leakage vulnerability in Android app logs. In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE 2024) (pp. 2510–2513). https://doi.org/10.1145/3691620.3695609.
16. OWASP Foundation. (2024). M2: Inadequate supply chain security. Retrieved from: https://owasp.org/www-project-mobile-top-10/2023-risks/m2-inadequate-supply-chain-security (date accessed: November 15, 2024).
17. NowSecure. (n.d.). Mobile app vetting for supply chain risk management. Retrieved from: https://www.nowsecure.com/solutions/by-need/mobile-app-vetting/ (date accessed: October 28, 2024).

18. Steinböck, M., Bleier, J., Rainer, M., Urban, T., Utz, C., & Lindorfer, M. (2024). Comparing Apples to Androids: Discovery, retrieval, and matching of iOS and Android apps for cross-platform analyses. In MSR '24: Proceedings of the 21st International Conference on Mining Software Repositories (pp. 348–360). https://doi.org/10.1145/3643991.3644896.

19. Alzubaidi, A. (2024). Detecting Android malware using deep learning algorithms: A survey. Computers and Electrical Engineering, 119(Part A), Article 109544. https://doi.org/10.1016/j.compeleceng.2024.109544.

20. Guardsquare. (2024). Assessing Mobile Application Security. Retrieved from: https://www.guardsquare.com/hubfs/Website/Resources/Reports/Guardsquare_Infographic-Assessing%20Mobile%20Application%20Security.pdf (date accessed: November 6, 2024).

Citation: Aleksandr Pinaev, "Ensuring the Security of the Mobile Application Supply Chain in the Context of the Evolution of Modern Mobile Threat Models", Universal Library of Innovative Research and Studies, 2024; 1(2): 111-117. DOI: https://doi.org/10.70315/uloap.ulirs.2024.0102014.