Hardcoded credentials in Android apps: Service exposure and category-based vulnerability analysis

Olha Mykhaylova[1,†], Taras Fedynyshyn[1,†] and Artem Platonenko[2,*,†]

[1] *Lviv Polytechnic National University, 12 Stepana Bandery str., 79013 Lviv, Ukraine*
[2] *Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudriavska str., 04053 Kyiv, Ukraine*

Abstract

This paper presents an extensive study of the security vulnerabilities in Android applications related to the hardcoding of sensitive credentials. A total of 6,165 APK files were downloaded from the Google Play Store and subjected to static analysis using Mobile Security Framework (MobSF). For each application, the "secrets" section, as identified by MobSF, was further examined using Trufflehog to detect and verify the presence of hardcoded credentials. The findings reveal a concerning prevalence of hardcoded credentials, with a significant portion of applications embedding sensitive information such as API keys and authentication tokens. The analysis identified various services for which credentials are frequently hardcoded, including cloud service providers, payment gateways, and third-party APIs. We also categorized the occurrence of hardcoded secrets by app type, analyzing the percentage of applications with exposed credentials across various Google Play categories. This study underscores the critical security risks posed by hardcoding secrets in mobile applications and provides insights into the scope and distribution of this vulnerability within the Android ecosystem. The results emphasize the need for stronger security practices in mobile app development, particularly regarding the secure management of sensitive information, and highlight potential areas of improvement in mobile application security.

Keywords

android security, mobile security, data privacy, static analysis, improper credentials usage, OWASP Mobile, MobSF, Trufflehog

1. Introduction

Mobile devices, particularly smartphones, have undergone constant evolution and are now the most common means for individuals to connect with others through phone calls or the Internet. Beyond communication, activities such as document handling, video streaming, emailing, and gaming can also be easily performed on smartphones, making them more versatile and essential than ever. According to [1], smartphones are expected to remain dominant, especially with the advent of 5G and future 6G.

Smartphones and the numerous applications that support various functions have become integral to modern life. Individuals increasingly depend on mobile applications for a wide range of daily tasks, utilizing them multiple times per day. The Apple App Store [2] and Google Play Store [3] offer over eight million applications combined. However, the provenance and security of these applications cannot always be guaranteed. Despite the vetting procedures employed by Apple and Google before allowing apps into their respective stores, many mobile applications still exhibit vulnerabilities and pose significant security risks. Notably, the data processed by these applications and mobile devices are frequent targets for cybercriminals. Mobile operating systems lack adequate tools to detect malware that can compromise personal data. As a result, mobile applications present potential security threats, as vulnerabilities within them may be exploited by attackers to gain unauthorized access to device resources, including sensitive user information [4].

Therefore, mobile applications are a vital element of the mobile ecosystem that necessitates further research to develop effective security methods and tools aimed at mitigating the risks associated with their use.

This study aims to conduct a large-scale static analysis of 6000+ Android applications from Google Play to identify and evaluate the presence of hardcoded sensitive information, such as API keys and credentials, using MobSF and Trufflehog. By detecting and analyzing these secrets, the study seeks to assess the security practices of mobile app developers, highlight potential vulnerabilities, and provide insights into improving the management of sensitive data within Android apps [5].

2. Background and related work

2.1. OWASP mobile Top 10

The Open Web Application Security Project (OWASP) is a nonprofit organization dedicated to enhancing software

CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, October 26, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
olha.o.mykhailova@lpnu.ua (O. Mykhaylova); fedynyshyn.taras@gmail.com (T. Fedynyshyn); a.platonenko@kubg.edu.ua (A. Platonenko)
0000-0002-3086-3160 (O. Mykhaylova); 0009-0006-8233-8057 (T. Fedynyshyn); 0000-0002-2962-5667 (A. Platonenko)

security through global collaboration and participation. OWASP provides a platform for leaders in industry, academia, and government to discuss and promote best practices in computing. Among its initiatives is the maintenance of a list highlighting the Top 10 Mobile Risks to mobile applications. This list identifies key security threats, including risks to data, internal and external device communications, and other vulnerabilities in mobile applications.

The list of common cyber threats to mobile applications and their descriptions, as outlined by the 2024 OWASP Mobile Top 10:

- M1: Improper Credential Usage—threat agents exploiting hardcoded credentials and improper credential usage in mobile applications can include automated attacks using publicly available or custom-built tools.
- M2: Inadequate Supply Chain Security—refers to the failure to secure third-party components, services, or libraries integrated into mobile applications, which can introduce vulnerabilities and increase the risk of compromise throughout the software supply chain.
- M3: Insecure Authentication/Authorization—treat agents that exploit authentication and authorization vulnerabilities typically do so through automated attacks that use available or custom-built tools.
- M4: Insufficient Input/Output Validation—insufficient validation and sanitization of data from external sources, such as user inputs or network data, in a mobile application can introduce severe security vulnerabilities.
- M5: Insecure Communication—refers to the failure to properly secure the transmission of sensitive data between the mobile app and external entities, such as servers or other devices, leading to potential interception, tampering, or exposure of information.
- M6: Inadequate Privacy Controls—refers to the insufficient protection of users' data within a mobile application, leading to unauthorized access, exposure, or misuse of sensitive information such as location, contacts, or other private data.
- M7: Insufficient Binary Protection—refers to the lack of proper defenses against reverse engineering or tampering with the mobile app's binary code, which can allow attackers to modify, exploit, or redistribute the application maliciously.
- M8: Security Misconfiguration—refers to the improper configuration of security settings, permissions, and controls that can lead to vulnerabilities and unauthorized access.
- M9: Insecure Data Storage—refers to the inadequate protection of sensitive data stored on a mobile device, which can lead to unauthorized access, data breaches, or exposure if the storage mechanisms are not properly secured.
- M10: Insufficient Cryptography—threat agents who exploit insecure cryptography in mobile applications can undermine the confidentiality, integrity, and authenticity of sensitive information [6].

2.2. Related works

The vulnerabilities of mobile applications, such as authentication and authorization errors, data leakage, and their associated security risks—ranging from API vulnerabilities, weak authorization and authentication, client-side injection, poor server-side security, insecure data storage and transmission, improper session handling, to the use of flawed or insecure encryption algorithms—pose significant threats [7]. In today's digital environment, users often entrust their devices with sensitive information, including financial and medical data, presenting a major cybersecurity challenge for mobile application developers and providers. Cybercriminals frequently target the data processed by mobile applications and devices [8]. Additionally, the rise of mobile applications for the Internet of Things (IoT) has heightened the threat of wormhole attacks [9–13].

NowSecure's benchmark testing [14] revealed that 85% of the applications examined contained one or more security risks. Over 50% of the analyzed applications exhibited vulnerabilities that compromised data protection during transmission. Additionally, approximately one-third of the tested applications had issues related to their source code. Notably, Android applications were particularly prone to code vulnerabilities, which could expose them to reverse engineering and other potential threats.

According to [15], the most common security issues in mobile applications include improper platform usage, insecure data storage, insecure client-server communication, insecure authentication (e.g., traditional password authentication imposes numerous limitations and is no longer considered secure or user-friendly for mobile users, while biometric authentication has gained attention as a promising solution for enhancing mobile security), insecure authorization, inadequate data encryption, poor code quality, code tampering, reverse engineering vulnerabilities, and extraneous functionality. The advancement of modern mobile application development technologies necessitates the parallel evolution of methods and tools to ensure their security. For instance, forecasting mobile application security on time can help implement preventive measures to reduce vulnerabilities and security risks [16]. Currently, there is a clear tension between the increasing number of mobile applications in use, along with the growing responsibilities they bear, and the inadequacy of existing security methods and tools.

3. Materials and methods

For this study, a comprehensive dataset comprising 6165 APK files was compiled from the Google Play Store [3]. This dataset was meticulously selected to represent a diverse array of applications across different categories and popularity tiers, thus ensuring a broad and representative sample of the mobile application ecosystem. The collected APKs underwent static analysis using the Mobile Security Framework (MobSF) [17], an established tool for assessing mobile application security. MobSF was utilized to perform an in-depth static analysis of each APK, focusing on the identification of potentially sensitive information embedded within the application's code. The static analysis process involved extracting a designated “secrets” section for each APK, which enumerates potential hardcoded secrets, including API keys, authentication tokens, and other credentials. After the static analysis, the extracted “secrets” sections were subjected to further scrutiny using Trufflehog [18], a tool specialized in detecting secrets within codebases. Trufflehog was employed to validate the authenticity of the identified secrets and to discern genuine secrets from false positives. This secondary analysis aimed to provide a more precise evaluation of the potential security risks associated with hardcoded credentials in the APKs. This methodological framework facilitated a rigorous examination of credential management practices within mobile applications and offered valuable insights into the security implications of secret exposure in Android applications.

3.1. Sample selection

As of 2024, the Google Play Store hosts over 3.5 million applications [19]. Conducting a comprehensive assessment of all these applications would demand substantial server resources and considerable time. Consequently, this study focused on analyzing a subset of the most popular applications. The initial step involved evaluating the popularity of mobile applications. Data on app downloads, segmented by country and category, was obtained from SimilarWeb [20]. At the time of the research, we identified 59,108 unique applications across 57 categories and 96 countries. Subsequently, APK files for these applications were downloaded for analysis. Given the absence of a direct method to download APK files from the Google Play Store, third-party services such as APKCombo [18] were utilized. Due to limitations in storage and computational resources, and the availability of APK files on third-party services, we were able to download and analyze 6,165 APK files.

The number of downloaded applications per Google Play category is listed in Table 1 which only includes 22 categories where at least 15 APK files were downloaded.

Table 1

App Category ID	Number of downloaded APK’s
SPORTS	580
PARENTING	566
PHOTOGRAPHY	452
NEWS_AND_MAGAZINES	443
SOCIAL	438
TOOLS	437
ENTERTAINMENT	417
PRODUCTIVITY	352
COMMUNICATION	312
AUTO_AND_VEHICLES	280
PERSONALIZATION	274
BOOKS_AND_REFERENCE	246
DATING	202
MUSIC_AND_AUDIO	187
MAPS_AND_NAVIGATION	173
ART_AND_DESIGN	139
BUSINESS	123
BEAUTY	102
EDUCATION	69
MEDICAL	68
HEALTH_AND_FITNESS	28
LIFESTYLE	17

3.2. Static analysis

Static analysis using MobSF is an essential technique for evaluating the security of mobile applications. MobSF is a versatile, open-source tool designed for the static analysis of both Android and iOS applications, aimed at identifying potential security vulnerabilities and insecure coding practices. The process begins when an APK (Android Package Kit) file is submitted to MobSF. Due to the nature of mobile application development, APKs must be disassembled and decompiled to allow for thorough examination. MobSF employs tools such as APKTool [21] and jadx [22] to decompile the APK, transforming the compiled bytecode into a more accessible, human-readable format. This step is crucial as it breaks down the application into its constituent components, including the manifest file, resources, and code. Once the APK is decompiled, MobSF performs an in-depth analysis of the application’s code. The analysis focuses on several key areas: the detection of sensitive data exposure, the identification of insecure coding practices, and the discovery of known vulnerabilities. MobSF scans the code for hardcoded secrets, such as API keys, credentials, and tokens, which can pose significant security risks if exposed. Additionally, the tool evaluates the use of cryptographic algorithms and other security measures to ensure they are implemented correctly.

3.3. Secrets post-processing with Trufflehog

Trufflehog [18] is a specialized tool designed to identify sensitive information, such as API keys, credentials, and tokens, within codebases. Initially developed for Git repositories, Trufflehog has proven [23] valuable in various security contexts, including static analysis of mobile applications and other software projects. Trufflehog’s core functionality relies on two primary techniques: pattern matching and entropy-based analysis. The tool employs a set of predefined regular expressions and heuristics to detect patterns commonly associated with secrets. These patterns include a variety of credentials and tokens that are often embedded directly within the application code. By leveraging these patterns, Trufflehog is capable of identifying a broad range of sensitive information that might otherwise be overlooked. In addition to pattern matching, Trufflehog utilizes entropy-based analysis to assess the randomness of certain strings within the code. Strings with high entropy values are indicative of potential secrets, as they are less likely to occur by chance in non-sensitive data. This method enhances Trufflehog’s ability to detect secrets that may not conform to established patterns but still pose a risk of exposure. For each detected secret, Trufflehog provides detailed information on its location within the code, which facilitates targeted remediation efforts. Trufflehog’s integration with other static analysis tools, such as MobSF, further enhances its utility. By analyzing the “secrets” sections extracted by tools like MobSF, Trufflehog can verify the authenticity of these findings and assess which secrets are genuinely at risk. This integration provides a more comprehensive assessment of an application’s security posture.

4. Results

This study conducted a comprehensive security analysis of 6165 Android applications among 22 categories. The analysis was performed using a combination of tools, including MobSF and Trufflehog to identify improper credential usage according to the OWASP MobileTop 10 framework. The results of the vulnerability analysis provide valuable insights into the security posture of the selected applications.

4.1. Hardcoded secret services

The research uncovered credentials for a variety of services, and the frequency of each type of credential was recorded. Fig. 1 shows the number of revealed secrets per service. As we can see Twitter consumer key is the most popular hardcoded credential.

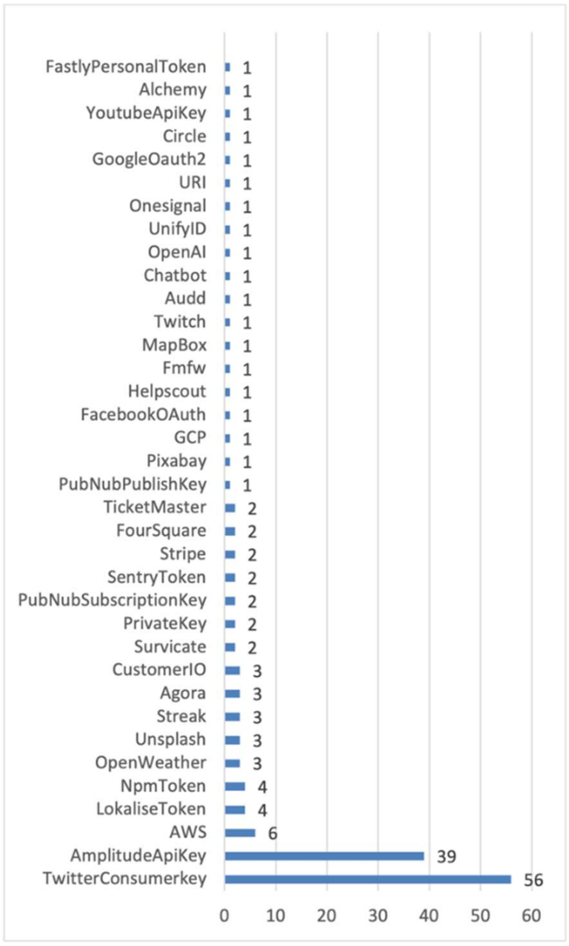

Figure 1: Number of found secrets per service

The research result shows some applications have cloud provider secrets hardcoded. Hardcoding AWS (Amazon Web Services) and GCP (Google Cloud Platform) secrets in mobile application code pose significant security risks, which can have serious implications for both the application and its users:

- Unauthorized Access and Data Breaches—hardcoded secrets, such as API keys and authentication tokens, provide direct access to cloud services and resources. If these secrets are exposed through the application code, malicious actors can exploit them to gain unauthorized access to cloud resources. This can lead to unauthorized data access, data breaches, and potential compromise of sensitive user information stored in the cloud [24].
- Increased Attack Surface—embedding secrets directly in the application code increases the attack surface, making it easier for attackers to identify and exploit vulnerabilities. Tools and techniques for reverse engineering can reveal these hardcoded secrets, allowing attackers to gain access to cloud services and escalate their attacks.
- Misuse of cloud resources—once an attacker obtains hardcoded cloud credentials, they can misuse cloud resources for malicious purposes. This might include launching unauthorized instances, executing costly operations, or conducting activities that could incur significant financial charges to the cloud account. This can lead to unexpected costs and resource depletion, affecting both the application's operation and its financial viability.
- Compromise of application integrity—hardcoded secrets may also lead to the compromise of application integrity. If attackers can exploit these credentials to modify or interfere with cloud services, they may alter application functionality, inject malicious code, or disrupt the normal operation of the app. This can undermine user trust and damage the application's reputation.
- Difficulty in rotation and management—hardcoded secrets complicate the management and rotation of credentials. Ideally, secrets should be regularly rotated and updated to reduce the risk of long-term exposure. However, hardcoded secrets require manual intervention to update, leading to potential lapses in security and prolonged exposure if credentials are compromised.
- Compliance and legal implications—hardcoding sensitive information in application code may also violate compliance regulations and legal requirements related to data protection and privacy. Regulations such as GDPR, HIPAA, and others mandate strict controls over the handling and protection of sensitive information. Exposing cloud credentials can result in non-compliance, legal repercussions, and fines.

4.2. Hardcoded secrets per app categories

The research demonstrates that applications in some Google Play categories have significantly different percentages of applications containing hardcoded secrets. Fig. 2 shows the number of scanned applications, the number of applications where secrets were detected, and the percentage of such applications per category.

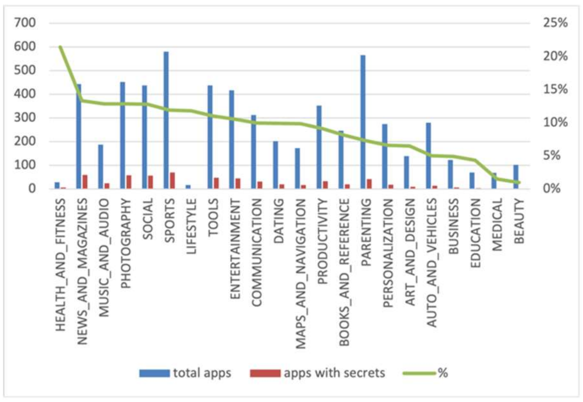

Figure 2: Percentage of apps with hardcoded secrets per category

As Fig. 2 shows category with the largest number of hardcoded secrets is "Health and fitness"—21% of applications in this category have hardcoded secrets. In the next four categories—"News and magazines", "Music and audio", "Photography" and "Social" 12% of applications have hardcoded credentials.

The important finding is that 10% of applications in the category "Communication" have hardcoded secrets. Hardcoding secrets such as credentials and API tokens in communication applications pose a variety of significant security risks, which can lead to severe consequences for both users and service providers. Communication apps, being highly sensitive due to their role in handling personal messages, calls, and media, are particularly vulnerable to attacks when secrets are embedded in the application code. Below are described some of the primary risks associated with hardcoding secrets in such applications:

- Unauthorized access to user data—hardcoded credentials can be easily extracted by attackers using reverse engineering techniques. This unauthorized access to API tokens or authentication keys may enable malicious actors to intercept sensitive user data, including personal messages, call logs, and media files. Such breaches present substantial privacy risks, as compromised data may be used for identity theft, surveillance, or exploitation.
- Compromise of communication integrity—the integrity of communication services depends on secure transmission channels. Exposed hardcoded secrets undermine this integrity, allowing attackers to impersonate legitimate users or services. This creates opportunities for man-in-the-middle (MITM) attacks, where communications may be intercepted, altered, or injected with malicious content without user awareness, jeopardizing the authenticity and confidentiality of the exchanged information.
- Service disruption and denial of service (DoS) attacks—attackers with access to hardcoded secrets may exploit them to abuse communication services by sending an excessive volume of requests or misusing APIs. Such actions can lead to Denial of Service (DoS) attacks, disrupting services for legitimate users. This type of attack not only impacts user experience but can also damage the service provider's reputation.
- Account takeover and identity theft—hardcoded API tokens or credentials allow attackers to take control of user accounts. This results in unauthorized access, where malicious actors can lock users out of their accounts, send fraudulent messages, or perform unauthorized actions. Account takeovers can lead to identity theft, social engineering attacks, or the dissemination of harmful content through compromised accounts.

5. Conclusions

This study provides a comprehensive examination of the prevalence and risks associated with hardcoded credentials within Android applications, highlighting a critical security gap in mobile application development.

By analyzing 6,165 Android applications across various categories using MobSF and Trufflehog, the research revealed that a significant number of applications contain hardcoded secrets, which pose substantial risks to user data privacy and application integrity. The findings indicate that hardcoded cloud provider secrets, such as AWS and GCP credentials, are common, representing a serious vulnerability that may lead to unauthorized access, resource misuse, and potential data breaches. Specifically, unauthorized access to sensitive data, compromise of application integrity, and increased exposure to Denial-of-Service (DoS) attacks were identified as potential consequences.

Additionally, hardcoded secrets complicate the rotation and management of credentials, making it difficult for developers to adhere to best practices for secure application management. Applications in categories such as Health and Fitness, News and Magazines, Music and Audio, Photography, and Social were particularly prone to containing hardcoded secrets, with Health and Fitness applications exhibiting the highest occurrence. Notably, communication applications were also found to have a high prevalence of hardcoded secrets, posing unique risks due to their handling of sensitive personal information, including messages, calls, and media.

This research underscores the urgent need for mobile developers to adopt secure coding practices, particularly in credential management, to reduce the risk of data breaches and protect user privacy. Implementing secure storage solutions for sensitive information and regular auditing of code for potential hardcoded credentials should become standard practices within the industry.

Additionally, frameworks and libraries should offer stronger guidance or automated tools for managing secrets to mitigate the risks associated with credential exposure.

Future work could focus on expanding this analysis to examine the impact of hardcoded secrets on user behavior and engagement, or on developing automated tools to detect and mitigate the risks associated with these vulnerabilities in real-time. This study ultimately reinforces the importance of secure credential handling as a fundamental aspect of mobile application security.

References

[1] C. Liu, et al., MobiPCR: Efficient, accurate, and strict ML-based mobile malware detection, Future Generation Comput. Syst. 144 (2023) 140–150. doi: 10.1016/j.future.2023.02.014.

[2] Apple Appstore. URL: https://www.apple.com/app-store/

[3] Google Play. URL: https://play.google.com/store

[4] O. Mykhaylova, et al., Mobile Application as a Critical Infrastructure Cyberattack Surface, in: Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3550 (2023) 29–43.

[5] Y. Dreis, et al., Model to Formation Data Base of Internal Parameters for Assessing the Status of the State Secret Protection, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 277–289.

[6] A. Horpenyuk, I. Opirskyy, P. Vorobets, Analysis of Problems and Prospects of Implementation of post-Quantum Cryptographic Algorithms, in: Classic, Quantum, and Post-Quantum Cryptography, vol. 3504 (2023) 39–49.

[7] E. Zaitseva, et al., Identifying the Mutual Correlations and Evaluating the Weights of Factors and Consequences of Mobile Application Insecurity, Systems, 11(5) (2023). doi: 10.3390/systems11050242.

[8] P. Zhu, et al., Using Blockchain Technology to Enhance the Traceability of Original Achievements, IEEE Trans. Eng. Manag. 70 (2023) 1693–1707.

[9] S.-Y. Kuo, F.-H. Tseng, Y.-H. Chou, Metaverse Intrusion Detection of Wormhole Attacks based on a Novel Statistical Mechanism, Future Gener. Comput. Syst. 143 (2023) 179–190.

[10] B. Zhurakovskyi, et al., Secured Remote Update Protocol in IoT Data Exchange System, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 67–76.

[11] V. Sokolov, et al., Method for Increasing the Various Sources Data Consistency for IoT Sensors, in: IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PICST) (2023) 522–526. doi: 10.1109/PICST57299.2022.10238518

[12] O. Shevchenko, et al., Methods of the Objects Identification and Recognition Research in the Networks with the IoT Concept Support, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 2923 (2021) 277–282.

[13] V. Dudykevych, et al., Platform for the Security of Cyber-Physical Systems and the IoT in the Intellectualization of Society, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 449–457.

[14] A Decade in, How Safe Are Your iOS and Android Apps? URL: https://www.nowsecure.com/blog/2018/07/11/adecade-in-how-safe-are-your-ios-and-android-apps

[15] Understanding OWASP Mobile Top 10 Risks with Real-World Cases. URL: https://appinventiv.com/blog/owaspmobile-top-10-real-world-cases/

[16] S. Shevchenko, et al., Protection of Information in Telecommunication Medical Systems based on a Risk-Oriented Approach, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 158–167.

[17] Mobile Security Framework (MobSF). URL: https://mobsf.github.io/docs/#/

[18] Trufflehog. URL: https://github.com/trufflesecurity/trufflehog

[19] How Many Apps in Google Play Store? (2024). URL: https://www.bankmycell.com/blog/number-of-google-play-store-apps/

[20] Similarweb Digital Intelligence: Unlock Your Digital Growth. URL: https://www.similarweb.com/

[21] Apktool – A Tool for Reverse Engineering Android APK Files. URL: https://apktool.org/

[22] jadx – Dex to Java Decompiler. URL: https://github.com/skylot/jadx

[23] S. K. Basak, et al., A Comparative Study of Software Secrets Reporting by Secret Detection Tools, 2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), New Orleans, LA, USA (2023) 1–12. doi: 10.1109/ESEM56168.2023.10304853.

[24] O. Deineka, et al., Designing Data Classification and Secure Store Policy According to SOC 2 Type II, in: Cybersecurity Providing in Information and Telecommun. Systems, vol. 3654 (2024) 398–409.