2nd International Conference of Information Technology to enhance E-learning and other Application, (IT-ELA2021),
Baghdad college of economic sciences University, Baghdad, Iraq

Android Mobile Applications Vulnerabilities and Prevention Methods: A Review

Hilmi Abdullah
Amedi Technical Institute
Duhok Polytechnic University
Duhok, Iraq
hilmi.salih@dpu.edu.krd

Subhi R. M. Zeebaree
Culture Center
Duhok Polytechnic University
Duhok, Iraq
subhi.rafeeq@dpu.edu.krd

2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA) | 978-1-6654-0952-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/IT-ELA52201.2021.9773615

***Abstract*— The popularity of mobile applications is rapidly increasing in the age of smartphones and tablets. Communication, social media, news, sending emails, buying, paying, viewing movies and streams, and playing games are just a few of the many uses for them. Android is currently the most popular mobile operating system on the planet. The android platform controls the mobile operating system market, and the number of Android Mobile applications grows day by day. At the same time, the number of attacks is also increasing. The attackers take advantage of vulnerable mobile applications to execute malicious code, which could be harmful and access sensitive private data. Security and privacy of data are critical and must be prioritized in mobile application development. To cope with the security threats, mobile application developers must understand the various types of vulnerabilities and prevention methods. Open Web Application Security Project (OWASP) lists the top 10 mobile applications security risks and vulnerabilities. Therefore, this paper investigates mobile applications vulnerabilities and their solutions.**

Keywords—Mobile Applications, Android Security, Open Web Application Security Project (OWASP), Vulnerabilities.

I. INTRODUCTION

Nowadays, people are using mobile applications more than ever, and their lifestyle is more dependent on them for shopping, entertainment, communication and many other areas. With the importance of these applications in our lives, the developers focus on improving the user experience at the expense of security. Mobile users leave a vast amount of sensitive and personal data in mobile devices, such as banking, health, photos, contacts and messages. When permission is granted to mobile applications to access services and data, there will be a potential risk of data gathering by untrustable parties that could harm the user [1]-[4].

Android is the Operating system for more than 2 billion devices [5], [6]. It is forecasted that 228.2 billion mobile applications will be downloaded by 2022 [1], [7]. Dealing with security issues of android mobile applications is a challenging and complex task [8], [9]. At the same time, the lack of security awareness among developers increases security vulnerabilities day by day. As a result, mobile application vulnerabilities research has become an insistent necessity [10].

In general, discovering vulnerabilities in Android applications can be via static and dynamic analysis [11]. Static analysis is achieved by restoring the application's source code, and then the vulnerabilities are identified in the source code. On the other hand, the application is loaded dynamically with dynamic analysis and simulated hacking attacks to detect vulnerabilities [12]-[14].

This paper studies various vulnerabilities of android applications and their mitigation techniques. The paper is organized as follows. Section II, presents the background of Android and its components. Section III, provides a literature review of the android applications and the security architecture of the applications. Section IV, discusses different types of vulnerabilities and how to prevent them. Also, a comparison is made among the reviewed researches. The assessment and recommendations are provided in the end of this section. Finally, the conclusion is provided in section V.

II. BACKGROUND THEORY

A. *Android*

Android is a Linux based operating system for smart devices. Android is open source and powers phones, tablets, cars and watches. The Linux kernel interacts with the hardware. On top of Linux, there are libraries and Dalvik Virtual Machine, which uses Linux core features such as multi-threading and memory management. The top layer of the Android operating system is the applications and application framework. Mobile applications interact with the phone via APIs [15]-[17].

B. *Android Run Time*

Android Run Time(ART) is a successor to Dalvik Virtual Machine which is used to execute applications written for Android. It takes the Android application and turns its java code to bytecode to be capable of running on a Linux system [17]-[19].

C. *Android Components*

Generally, there are four types of components in Android. Mobile applications could be composed on all of them are less depending on the complexity of the application. Below are the four types of components in Android [15], [16], [20]:

Activity: This component is responsible for handling user interaction with the screen of the smartphone.
Services: Services are responsible for the background processing associated with the mobile application.
Broadcast Receiver: The communication between the Android OS and the applications is handled by broadcast receivers.
Content Providers: They are responsible for handling the data and database management.

III. Literature Review

There has been a lot of research in the field of mobile applications security. According to [21], [22], the number of vulnerabilities of Android Applications continues to grow. These vulnerabilities cause various types of attacks such as denial of service, sniffing, privilege escalation and non-authenticated access. Many participants contribute to the vulnerability market consisting of vendors, Mobile Applications, vulnerabilities, and hunters. The vendors release mobile applications, and the hunters discover the potential vulnerabilities. Hunters might take advantage of these vulnerabilities and sell them in the black market, which will cause damage to the consumers. It is also possible that hunters will inform the vendors to fix the vulnerabilities [23].

Android Applications can be downloaded from Google Play as well as third-party sites. The downloaded applications are compressed as Android Packages (APK), which includes libraries, metadata and files necessary for the application to be installed and run. In order to protect the privacy of users and prevent malicious acts, Android enforces applications to request permissions from users to access sensitive information. Thus, the efficiency of the permission mechanism, together with the users' understanding, plays an essential role in Android security [24], [25]. However, Android applications suffer from various threats such as intent spoofing, privilege escalation, phishing and others [23] .

In general, there are various causes for android vulnerabilities, such as libraries and the outdated version of android. According to [1], [8], [26], around 10% of mobile devices are running on Android 4.3 or lower, which means that millions of devices are vulnerable. These devices need to upgrade to the latest version of android to fix vulnerabilities. Moreover, another cause is the vulnerable libraries, as [27] stated that more than 50% percent of vulnerabilities in Android applications are due to vulnerable libraries.

The android security architecture consists of the following:

- Permission Mechanism: By this mechanism, security interfaces such as Phone calls and SMS are protected, meaning that the application needs to have permission to perform an operation such as sending SMS [28].
- Sandboxing is a method to isolate application resources from system resources and from the components of other applications on the same device, as in fig. 1 [28], [29].
- Access control: with this mechanism, there are a set of rules for reading, writing and executing files by processes [28].
- *Components Encapsulation: the components of android applications are encapsulated by declaring them as private or public, which determines how these components can be accessed [28].*
- *Application Signing: Android applications are signed by the developers with a cryptographic signature. This certificate identifies the developer as the author [28].*

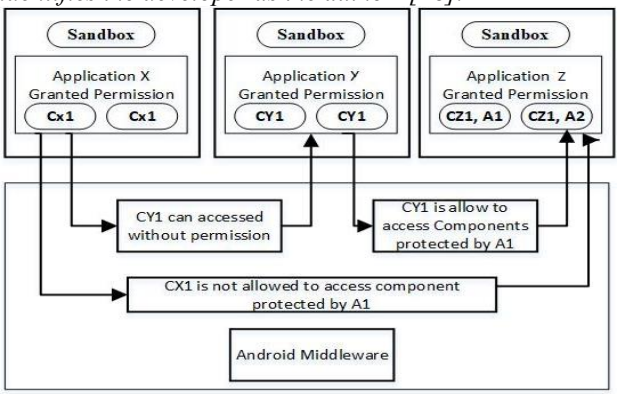

Fig 1. Mobile Application Sandboxing

When developing mobile applications, it is important to maintain their security, and the mobile application developers should be aware of the vulnerabilities and how to mitigate them. Open Web Application Security Project (OWASP) is a foundation that focuses on Software and Mobile applications security and lists the vulnerabilities and their mitigation. Below is the list of OWASP top 10 mobile applications vulnerabilities:

A. Weak Server-Side Controls

Even though most mobile applications are similar to client/server applications that depend on a connection with the server, many mobile application developers do not consider the security concerns of the traditional server-side applications.

Solution: To prevent Weak Server-Side Controls, it is required to scan mobile applications with scanning tools to identify any security vulnerabilities. Generally, automated scanning tools are used. However, manually assessing is sometimes needed to detect any vulnerability which automated scanning tools haven't detected. In addition, it is recommended that mobile application developers follow secure coding standards and practices to reduce security vulnerabilities[21], [22].

B. Insecure Data Storage

Typically, mobile applications developers store client-side personal or sensitive data in databases and files. When these data are stored without a secure mechanism, attackers can get access these data by rooting or jailbreaking the mobile device. Several problems are associated with insecure data storage, such as fraud and identity or credential theft.

Solution: As a common rule, sensitive data should not be stored in mobile applications unless necessary. For instance, the users should use APIs to authenticate instead of storing their credentials on the phone's file system. Furthermore, all local

files should be encrypted, and the encryption keys should be stored securely [21], [22].

C. Insufficient Transport Layer Protection

This vulnerability means that the network traffic is not protected when data is transferred between the mobile application and the backend.

Solution: The solution to Insufficient Transport Layer Protection is to use TLS to transmit critical information and session tokens. In addition, by utilizing trusted certificates and pinning them into the mobile application service [8], [29].

D. Unintended Data Leakage

Unintended Data Leakage happens when the mobile application developer accidentally stores sensitive data in a location accessible to other mobile applications running on the same device.

Solution: To prevent this vulnerability, it is necessary to block caching and clipboard access [8], [30].

E. Poor Authorization and Authentication

This vulnerability happens when attackers get access to sensitive mobile application data by bypassing the authentication protocol. When mobile applications work offline, they are more vulnerable to these attacks.

Solution: This vulnerability is to ensure that authentication is done on the server-side instead of the client-side and in the offline mode, assigning fewer privileges to mobile application users to avoid unauthorized access to sensitive data [21].

F. Broken Cryptography

Adopting a faulty or hazardous cryptographic algorithm is an unnecessary risk that could lead to exposed sensitive data. Adopting a non-standard algorithm is risky because a determined attacker may be able to decipher the algorithm and gain access to the protected data. Cryptography is a critical part of data security. Many mobile application developers hardcode the encryption and decryption keys in the application's source code, making them vulnerable to attackers via reverse engineering. Moreover, some cryptographic algorithms are weak and can't protect mobile apps like SHA1 and MD5.

Solution: The solution to broken cryptography vulnerability is to stick to the cryptographic standards and never store sensitive data unless necessary [29].

G. Client-Side Injection

The client-side injection causes malicious code to be executed via the mobile app via the mobile device. This malicious code is typically delivered in the form of data that the threat agent sends to the mobile app via various methods. Generally, malicious code is injected by attackers via input data. This code might compromise the system with privileged permissions when it is executed.

Solution: To prevent code injection, the data must be validated, and parameterized queries must be used when accessing the database [22], [31].

H. Security Decisions Via Untrusted Inputs:

In the development process of a mobile application, hidden files or values are usually used to differentiate between different types of users. When attackers intercept these data, they can obtain unauthorized access and privilege permissions.

Solution: In the development process, security protocols should be followed and dealing with URL schemes with caution [22].

I. Improper Session Handling

For session management in mobile applications, developers use OAUTH tokens, SSO and cookies. When attackers gain access to session tokens, they can carry out transactions the same as legitimate users and gain access to higher privileges.

Solution: To avoid this vulnerability, proper session handling must be implemented by authenticating the user via backend and using session tokens that are difficult to guess and configuring session timeouts [5], [32].

J. Lack of Binary Protections

This vulnerability can expose sensitive mobile application data such as credentials. The attackers can use reverse engineering to find such sensitive information as in Fig. 2 [28].

Solution: To prevent this vulnerability, sensitive API keys should be stored on the server, and passwords shouldn't be stored in mobile applications binary2 [28].

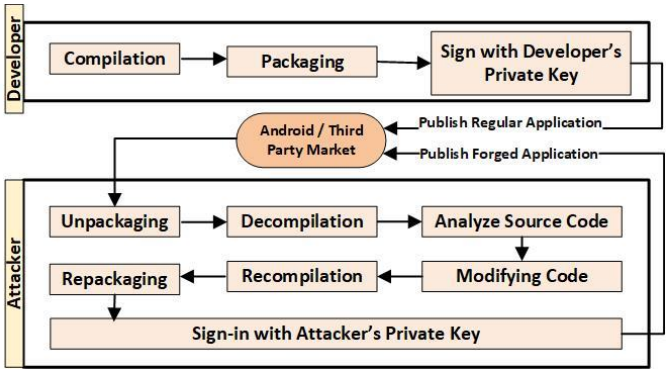

Fig 2. Android Application Forgery Process

IV. DISCUSSION, COMPARISON, ASSESSMENT AND RECOMMENDATION

A. Discussion and Comparison

This section provides a sufficient discussion for the previous works explained in the literature review section. Table I illustrates concentrated comparison among the features of all these previous works. Also, a specific assessment and recommendation will be presented after the comparison table to help the next researchers who want to work in this field.

We studied various android mobile application security researches and summarized each research's findings, as shown in Table I. The researchers used various methodologies in their research, such as reverse engineering, case studies, and

developing applications for the research purpose. According to [18], [33], lack of security awareness and bad coding practices among the developers when developing mobile applications lead to vulnerabilities in the applications. On the other hand, [24], [25], [29], [34], [35] stated that the permission system contains flaws and the improper usage of the Android permission system lead to vulnerabilities and data leakage. Furthermore, according to [8], [16], certificate validation plays an important role in enhancing the security of mobile applications. Finally, [8], [16], [36] stated that many mobile application vulnerabilities occur due to incorrect usage of APIs.

TABLE I. COMPARISON OF REVIEWED RESEARCHES

Author	Year	Objectives	Findings	Methodology
Visoottiviseth et al. [33]	2019	Mobile Application Security assessment	Lack of security awareness among developers.	Developing MASai application which finds vulnerabilities
Almomani and Al Khayer [25]	2020	Analysis of Android Permission system	The permission system contains flaws that might compromise the operating system.	Case study of applications permissions changes in different Application versions.
Ngobeni and Mhlongo [24]	2019	Enhancing mobile user awareness	A lack of knowledge about Android permissions can lead to users' private information being compromised by dangerous and malicious applications	Quantitative approach about user awareness of permissions
Kouraogo et al. [37]	2016	Studying of Attacks on Android Banking applications	Broken cryptography and reverse engineering	Reverse Engineering and static code analysis
Karthick and Binu [34]	2017	Android security issues and solutions	Misuse of Applications permissions using Shared User ID	Developing an application that tracks shared User ID
Bhor and Karia [38]	2017	Certificate pinning in Android applications	Exchanging unencrypted data between the mobile application and the server	Sniffing the network traffic with SSL decryption
Rahman et al. [18]	2017	Analysis of static code metrics to predict risk	Bad Coding Practices	Static code analysis to predict application defects
Schmeelk and A. Aho [8], [16]	2017	Android applications availability	Risky API Calls and Risky Libraries	A watchdog application to discover availability concerns
Qin et al. [8], [16]	2020	Android Vulnerabilities Related with Web Functions	Improper certificate validation	Case study for a large number of Android applications
Gong et al. [39]	2020	Predicting Android application security risk based on code smells	Code smells such as member ignoring method (MIM) and leaking inner class (LIC) were more important in predicting security risk levels.	Applying Machine learning algorithms to code smells and metrics
Mos and Chowdhury [40]	2020	Studying android threats and protection methods	Installing trustworthy applications and treat permissions carefully	Studying and classification of threats
chao et al. [11]	2020	Mining Vulnerabilities in Android applications	Component exposure and weak certificated check are the most detected vulnerabilities	Implementing a static and dynamic vulnerability mining engine
Song et al. [41]	2018	JavaScript related Vulnerabilities in Android applications	Many JavaScript exploitable vulnerabilities can lead to successful real attacks	I am developing a static analysis tool that detects Android Apps' JavaScript-related vulnerabilities.
Tang et al [36]	2020	Tang et al. Analyzing security vulnerabilities in Android mobile applications	Many vulnerabilities are due to incorrect usage of APIs	Static analysis and designing dynamic executable scripts.
Yang et al. [35]	2021	Privacy Risk Assessment for Android Applications	Improper usage of permissions could lead to data leakage	Implementing a framework PRADroid for privacy risk assessment
Pande et al.[32]	2021	They were providing security to android applications via confinement malicious applications.	A single mandatory access control policy governs all android applications belonging to the same domain.	Proposed a security scheme to limit unauthorized access of resources by android applications.
Zhan et al. [26]	2021	Identifying vulnerabilities in third-party libraries in android applications	Vulnerable third party libraries used in the development of android applications.	Proposed ATVHunter system, which identifies the precise vulnerable third party libraries in the application.

B. Assessment and ecommenation

Android application vulnerabilities have become a serious concern. Many studies have been performed to address these vulnerabilities and find solutions to mitigate their risks. In this study, we have reviewed many articles that analyse android mobile application vulnerabilities with various methodologies. According to [17], [24], [25] and the vulnerabilities are mainly due to the Android permission system ranging from flaws to the lack of knowledge and misuse of application permissions. Moreover, [15], [16] have shown that data encryption plays a

critical role in preventing flaws that could lead to serious privacy issues. Furthermore, [18], [22] have emphasized the importance of considering security issues by developers at the development phase of mobile applications.

In general, to achieve a good level of security in Android, there should be user awareness on understanding permissions and the importance of updates. At the same time, mobile application developers should consider security issues carefully. In addition, encryption and storage of sensitive data should be taken care of securely.

V. Conclusion

From the reviewed research, it can be concluded that this field's focus should be on the lack of security awareness among developers. In addition, a lack of understanding of Android permissions can lead to harmful and malicious applications gaining access to users' personal information and data. Android-based mobile applications' security has become a serious issue because of various types of vulnerabilities. These vulnerabilities might have a critical impact on the users and lead to serious problems if not handled correctly. In this paper, we study the vulnerabilities and the methods of mitigation of the threats. There were various types of threats that were mostly related to android permission system and mobile user awareness and lack of security knowledge in the mobile application development process. This paper aims to address the risks to help the mobile applications developers focus on security and privacy during the development process.

References

[1] A. e Brilingait\.e, L. Bukauskas, and E. Kutka, "Detection of Premeditated Security Vulnerabilities in Mobile Applications," in *ECCWS 2019 18th European Conference on Cyber Warfare and Security*, 2019, p. 63.

[2] S. Zeebaree, "DES encryption and decryption algorithm implementation based on FPGA," *Indones. J. Electr. Eng. Comput.* Sci, vol. 18, no. 2, pp. 774–781, 2020.

[3] S. R. Zeebaree, H. M. Shukur, and B. K. Hussan, "Human resource management systems for enterprise organizations: A review," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 7, no. 2, pp. 660–669, 2019.

[4] S. M. Mohammed, K. Jacksi, and S. Zeebaree, "A state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 1, pp. 552–562, 2021.

[5] J. Gao, L. Li, P. Kong, T. F. Bissyandé, and J. Klein, "Understanding the evolution of android app vulnerabilities," *IEEE Transactions on Reliability*, 2019.

[6] Z. N. Rashid, S. R. Zeebaree, and A. Shengul, "Design and analysis of proposed remote controlling distributed parallel computing system over the cloud," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 2019, pp. 118–123.

[7] S. R. Zeebaree and H. Rajab, "Design and implement a proposed multi-sources to multi-destinations broadcasting video-signals," in *2019 4th Scientific International Conference Najaf (SICN)*, 2019, pp. 103–108.

[8] J. Qin, H. Zhang, J. Guo, S. Wang, Q. Wen, and Y. Shi, "Vulnerability Detection on Android Apps-Inspired by Case Study on Vulnerability Related With Web Functions," *IEEE Access*, vol. 8, pp. 106437–106451, 2020.

[9] Z. A. Younis, A. M. Abdulazeez, S. R. Zeebaree, R. R. Zebari, and D. Q. Zeebaree, "Mobile Ad Hoc Network in Disaster Area Network Scenario: A Review on Routing Protocols.," *International Journal of Online \& Biomedical Engineering*, vol. 17, no. 3, 2021.

[10] S. R. Zeebaree, A. B. Sallow, B. K. Hussan, and S. M. Ali, "Design and simulation of high-speed parallel/sequential simplified DES code breaking based on FPGA," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 2019, pp. 76–81.

[11] W. Chao, L. Qun, W. XiaoHu, R. TianYu, D. JiaHan, G. GuangXin, and S. EnJie, "An android application vulnerability mining method based on static and dynamic analysis," in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2020, pp. 599–603.

[12] Z. Zhou, C. Sun, J. Lu, and F. Lv, "Research and implementation of mobile application security detection combining static and dynamic," in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2018, pp. 243–247.

[13] S. V. Amanoul, A. M. Abdulazeez, D. Q. Zeebare, and F. Y. Ahmed, "Intrusion Detection Systems Based on Machine Learning Algorithms," in *2021 IEEE International Conference on Automatic Control \& Intelligent Systems (I2CACIS)*, 2021, pp. 282–287.

[14] Z. N. Rashid, S. R. Zebari, K. H. Sharif, and K. Jacksi, "Distributed cloud computing and distributed parallel computing: A review," in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, 2018, pp. 167–172.

[15] A. Nirumand, B. Zamani, and B. Tork Ladani, "VAnDroid: A framework for vulnerability analysis of Android applications using a model-driven reverse engineering technique," *Software: Practice and Experience*, vol. 49, no. 1, pp. 70–99, 2019.

[16] S. Schmeelk and A. Aho, "Defending android applications availability," in *2017 IEEE 28th Annual Software Technology Conference (STC)*, 2017, pp. 1–5.

[17] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security," in *Proceedings of the 20th USENIX security symposium*, 2011, no. August.

[18] A. Rahman, P. Pradhan, A. Partho, and L. Williams, "Predicting Android application security and privacy risk with static code metrics," in *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2017, pp. 149–153.

[19] Z. S. Hammed, S. Y. Ameen, and S. R. Zeebaree, "Massive MIMO-OFDM Performance Enhancement on 5G," in *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2021, pp. 1–6.

[20] I. Khokhlov and L. Reznik, "Android system security evaluation," in *2018 15th IEEE Annual Consumer Communications \& Networking Conference (CCNC)*, 2018, pp. 1–2.

[21] D. Yadav, D. Gupta, D. Singh, D. Kumar, and U. Sharma, "Vulnerabilities and security of web applications," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, 2018, pp. 1–5.

[22] G. Koala, D. Bassolé, A. Zerbo, T. F. Bissyandé, O. Sié, and others, "Analysis of the Impact of Permissions on the Vulnerability of Mobile Applications," in *International Conference on e-Infrastructure and e-Services for Developing Countries*, 2019, pp. 3–14.

[23] Y. Wang, Y. Zhang, K. Wang, and J. Yan, "Security Analysis and Vulnerability Detection of Gesture-Based Lock in Android Applications," in *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 410–417.

[24] A. Ngobeni and S. Mhlongo, "Towards Enhancing Security in Android Operating Systems-Android Permissions \& User Unawareness," in *2019 2nd International Conference on Computer Applications \& Information Security (ICCAIS)*, 2019, pp. 1–6.

[25] I. M. Almomani and A. Al Khayer, "A comprehensive analysis of the android permissions system," *IEEE Access*, vol. 8, pp. 216671–216688, 2020.

[26] X. Zhan, L. Fan, S. Chen, F. Wu, T. Liu, X. Luo, and Y. Liu, "Atvhunter: Reliable version detection of third-party libraries for vulnerability identification in android applications," in *2021 IEEE/ACM 43rd International Conference on Software Engineering*

(ICSE), 2021, pp. 1695–1707.

[27] T. Watanabe, M. Akiyama, F. Kanei, E. Shioji, Y. Takata, B. Sun, Y. Ishi, T. Shibahara, T. Yagi, and T. Mori, "Understanding the origins of mobile app vulnerabilities: A large-scale measurement study of free and paid apps," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 14–24.

[28] J. B. Hur and J. A. Shamsi, "A survey on security issues, vulnerabilities and attacks in Android based smartphone," in *2017 International Conference on Information and Communication Technologies (ICICT)*, 2017, pp. 40–46.

[29] S. H. Haji and S. Y. Ameen, "Attack and anomaly detection in iot networks using machine learning techniques: A review," *Asian Journal of Research in Computer Science*, pp. 30–46, 2021.

[30] G. Y. Izadeen and S. Y. Ameen, "Smart android graphical password strategy: A review," *Asian Journal of Research in Computer Science*, pp. 59–69, 2021.

[31] F. Q. Kareem, S. Y. Ameen, A. A. Salih, D. M. Ahmed, S. F. Kak, H. M. Yasin, I. M. Ibrahim, A. M. Ahmed, Z. N. Rashid, and N. Omar, "SQL injection attacks prevention system technology," *Asian Journal of Research in Computer Science*, pp. 13–32, 2021.

[32] P. Pande, K. Mallaiah, R. K. Gandhi, A. K. Medatiya, and S. Srinivasachary, "Fine Grained Confinement of Untrusted Third-Party Applications in Android," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021, pp. 372–376.

[33] V. Visoottiviseth, C. Kotarasu, N. Cheunprapanusorn, and T. Chamornmarn, "A Mobile Application for Security Assessment Towards the Internet of Thing Devices," in *2019 IEEE 6th Asian Conference on Defence Technology (ACDT)*, 2019, pp. 1–7.

[34] S. Karthick and S. Binu, "Android security issues and solutions," in *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2017, pp. 686–689.

[35] Y. Yang, X. Du, and Z. Yang, "PRADroid: Privacy Risk Assessment for Android Applications," in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, 2021, pp. 90–95.

[36] J. Tang, R. Li, K. Wang, X. Gu, and Z. Xu, "A novel hybrid method to analyze security vulnerabilities in android applications," *Tsinghua Science and Technology*, vol. 25, no. 5, pp. 589–603, 2020.

[37] Y. Kouraogo, K. Zkik, G. Orhanou, and others, "Attacks on Android banking applications," in *2016 International Conference on Engineering \& MIS (ICEMIS)*, 2016, pp. 1–6.

[38] M. Bhor and D. Karia, "Certificate pinning for android applications," in *2017 International Conference on Inventive Systems and Control (ICISC)*, 2017, pp. 1–4.

[39] A. Gong, Y. Zhong, W. Zou, Y. Shi, and C. Fang, "Incorporating Android Code Smells into Java Static Code Metrics for Security Risk Prediction of Android Applications," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, 2020, pp. 30–40.

[40] A. Mos and M. M. Chowdhury, "Mobile Security: A Look into Android," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, 2020, pp. 638–642.

[41] W. Song, Q. Huang, and J. Huang, "Understanding javascript vulnerabilities in large real-world Android applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 1063–1078, 2018.