**Mobile Application Vulnerability Scanner (Android Armor)**

**Adhithya.C (**CH.EN.U4CYS21002**), Diyanesh.M.S (**CH.EN.U4CYS21015**)**

*UG Scholar Amrita School of Computing, Amrita Vishwa Vidyapeetham – Chennai*

***Abstract:-*** Current static analysis tools for Android APK files often fall short in providing clear and actionable insights into vulnerabilities, leaving users perplexed by vague reports or false positives. This limitation hinders the proactive identification and mitigation of security risks in mobile applications, emphasizing the need for a tool like Android Armor with its unique focus on comprehensive explanations and practical exploitation guidance. Android Armor (aa.sh) is an Android APK static analysis tool with a distinct approach. It offers a "Verbose Analysis" (-v) option that explains identified vulnerabilities comprehensively, eliminating ambiguity. Additionally, it provides an "Exploit Guidance" (-e) feature, supplying users with precise terminal commands to either exploit or manually verify vulnerabilities, fostering a deeper understanding of security risks. Mavericks bridges the gap between detection and understanding, empowering security professionals and developers to proactively secure Android applications. This tool stands out by offering clarity in vulnerability reports and practical exploitation guidance, making it a valuable asset in mobile application security analysis.

**Keywords :** Android Armor, Verbose Analysis, Exploit Guidance, Static Analysis, Android Apk

1. INTRODUCTION

In today's rapidly expanding digital landscape, mobile applications have become an integral part of our daily lives. With over 2 billion apps available on the Apple App Store and 2.2 million on the Google Play Store, mobile app usage is at an all-time high. However, this surge in app development and usage has also given rise to a critical concern – the prevalence of high-risk security vulnerabilities within these applications. Recent research has highlighted the gravity of this issue, revealing that a significant 25% of mobile apps contain at least one significant security vulnerability. Even more concerning is the discovery that approximately 59% of finance apps on the Android platform exhibit three or more vulnerabilities from the OWASP Mobile Top 10 risks, a set of well-recognized security concerns in mobile applications. These vulnerabilities encompass a wide range of threats, posing risks to both user data and device security.

To combat these risks, the field of mobile app security employs various techniques, one of which is static analysis. Unlike dynamic detection methods that involve running the app and observing its behavior, static analysis involves a thorough examination of the app's code and resources without execution. This proactive approach allows security experts to identify potential threats and vulnerabilities before they can be exploited.In this paper, we introduce an innovative architecture tailored for vulnerability analysis in Android applications, which we have aptly named the "Vulnerability Parser." The Vulnerability Parser is designed to comprehensively analyze Android applications, uncovering potential security flaws.

To achieve this analysis, the Android Armor harnesses a suite of powerful tools:

* apkinfo: This tool is utilized to extract crucial information from Android application packages (APKs). It provides valuable insights into the app's structure, permissions, and certificates.
* d2j - dex2jar: Dex2jar converts Dalvik bytecode (the format Android apps are typically written in) into a more human-readable and analyzable format. This transformation is vital for in-depth code inspection.
* zipgrep: Zipgrep is employed to search for specific patterns and content within APK files, helping identify potential security issues hidden within the app's resources.
* apktool: Apktool is instrumental in the reverse-engineering process. It disassembles APK files, making it possible to examine the app's code, resources, and manifest file in a human-readable form.

By leveraging the capabilities of these tools, the Android Armor aims to provide a robust and effective means of identifying and addressing security vulnerabilities in Android applications. This architecture's detailed implementation will be further explored in the subsequent sections..

1. LITERATURE REVIEW

Numerous approaches and solutions have already been put forth for mobile application vulnerability , which has been the subject of extensive study. Below are some summarize of previously suggested solutions and their shortcomings.

The paper "Vulnerability Parser: A Static Vulnerability Analysis System for Android Applications" introduces a static analysis system for Android apps, aiming to identify security vulnerabilities. While it provides a valuable tool for enhancing app security, its drawback lies in the absence of quantitative results on its effectiveness and a lack of comparison with existing tools, potentially limiting its applicability and benchmarking against other vulnerability detection systems.

The paper "Android Mobile Applications Vulnerabilities and Prevention Methods: A Review" authored by Hilmi Salih Abdullah and Subhi R. M. Zeebar presents a comprehensive review of vulnerabilities in Android mobile applications and prevention methods. While it offers valuable insights into the subject, it lacks recent updates as it may not reflect the latest developments in mobile app security, potentially limiting its relevance for current security practices.

The paper Vulnerability Detection on Android Apps–Inspired by Case Study on Vulnerability Related With Web Functions highlights the growing dependence on mobile apps for various aspects of daily life and the associated security challenges. It presents a methodology for analyzing vulnerabilities stemming from API misuse in Android finance, shopping, and browser apps. The authors introduce VulArcher, a scalable tool, to detect four specific vulnerabilities in over 6000 sampled apps, achieving high accuracy rates and contributing to a national vulnerability database

The paper Mobile Applications -Vulnerability Assessment Through the Static and Dynamic Analysis by Sreenivasa Rao Basavala, Narendra Kumar, Alok Agarrwal addresses the increasing use of mobile applications across various sectors, emphasizing their critical need for security features like authentication, data protection, and privacy. Mobile apps, including those for banking, e-commerce, and healthcare, are becoming central to daily life, raising concerns about security vulnerabilities. The study highlights the challenges in testing and securing mobile apps and underscores that they are not inherently more secure than desktop or web applications, calling for rigorous security testing and mitigation measures.

The paper Penetration Frameworks and Development Issues in Secure Mobile Application Development: A Systematic Literature Review by IKRAM UL HAQ AND TAMIM AHMED KHAN explores the challenges faced by novice Android app developers who often lack awareness of the latest Android vulnerabilities. It evaluates existing frameworks and techniques using the ISO/IEC 25010 software quality model, highlighting the hurdles in designing secure Android applications. The paper also conducts a comprehensive survey of penetration tools, assessing their suitability for vulnerability modeling during the design phase. Ultimately, the study identifies critical issues and gaps, paving the way for the development of a framework or tool to enhance the security of Android apps through vulnerability integration during the design stage.

III. METHODOLOGY

This expanded methodology provides an even more detailed view of how the "Automated Mobile Application Vulnerability Scanner" operates, breaking down each step, sub-check, and aspect of analysis in a comprehensive manner This methodology encompasses a series of well-defined steps to ensure the identification of potential vulnerabilities within mobile applications. It aims to leave no stone unturned when it comes to identifying potential security vulnerabilities within Android APK files

This methodology explains 8 main division which further has some sub division. The major division are Information Extraction, JAR File Creation, Checking for Misconfigurations, Checking for Outdated Software, AndroidManifest.xml Examination Framework Detection , Insecure Mobile Device Data Storage and final one is Experimental Analysis which states the result of an apk scanned using Android Armor.

**3.1 Information Extraction:**

The scanning process begins with the extraction of essential information from the Android APK file. This information includes critical details about the application, such as:

* Application Name: The human-readable name of the mobile application.
* Package Name: A unique identifier for the application within the Android ecosystem.
* Version Name: The user-friendly version of the application.
* Version Code: A numerical representation of the application's version.

To accomplish this, the scanner utilizes the "apkinfo" tool, which parses the APK file and retrieves these vital details. Information extraction serves as the foundational step for subsequent analysis, allowing users and developers to gain insights into the scanned application.

**3.2 JAR File Creation:**

To delve into the code of the mobile application, the scanner creates a .jar (Java Archive) file from the original APK. This step is crucial for conducting in-depth code analysis. The conversion from APK to .jar format is carried out using the "d2j-dex2jar" utility, which specializes in translating Android's Dalvik bytecode into Java bytecode.

The resulting .jar file serves as the basis for identifying potential vulnerabilities in the application's source code, enabling further scrutiny of its inner workings.

**3.3 Checking for Misconfigurations:**

3.3.1 Insufficient Certificate Validation

One of the primary areas of concern in mobile application security is the validation of SSL/TLS certificates. Insecure certificate validation can lead to data interception by malicious actors. The scanner conducts the following checks to identify certificate-related vulnerabilities:

Hostname Verification - It searches for instances where ALLOW\_ALL\_HOSTNAME\_VERIFIER is used within the code. The presence of this constant indicates that hostname verification may be disabled, potentially exposing the application to man-in-the-middle attacks.

Auth in Protected Space - The scanner looks for the existence of canAuthenticateAgainstProtectionSpace. This function is critical for strong authentication in secure contexts. Its absence or improper usage can be indicative of authentication weaknesses.

3.3.2 Logging Enabled

Logging is essential for debugging and monitoring, but it can also pose a significant security risk if sensitive information is logged inadvertently. The scanner scrutinizes the application's code to detect instances of logging, focusing on:

* Log.e Calls: The presence of Log.e calls, which are typically used to log error messages. If sensitive data is included in these logs, it can be exposed in the event of a breach.
* Logger Calls: The scanner identifies instances where custom loggers are used, as they may also be employed to log sensitive information.

3.3.3 Snapshot and Backup Checks

Data snapshots and backups are essential features for mobile applications, but they can introduce security vulnerabilities if not properly configured. The scanner assesses the following aspects related to snapshots and backups:

Snapshots Allowed: It examines the Android Manifest .xml file to determine if the application allows snapshots by checking for the presence of excludeFromRecents = "true". Allowing snapshots could expose sensitive data.

Backups Allowed: In the AndroidManifest.xml file, it verifies whether the application allows backups by looking for android:allowBackup="true". Enabling backups without proper encryption and access controls can lead to data leaks.

**3.4 Checking for Outdated Software:**

The scanner recognizes that mobile applications may rely on third-party libraries, and these libraries may have their own vulnerabilities. To address this concern, the scanner searches for mentions of outdated software libraries within the APK, with a particular focus on outdated ' libpng ' versions. Outdated libraries can introduce known vulnerabilities that attackers may exploit.

**3.5 AndroidManifest.xml Examination:**

The AndroidManifest.xml file contains crucial configuration details about the application. The scanner conducts an in-depth analysis of this file to identify additional misconfigurations and vulnerabilities:

3.5.1 Backups Allowed

Continuing from the initial check for backups, the scanner reiterates its importance by examining the AndroidManifest.xml file to determine if backups are allowed. This additional check reinforces the significance of secure backup configurations.

3.5.2 Cleartext Allowed

Mobile applications should use secure communication channels to protect user data. The scanner inspects the AndroidManifest.xml file to detect the presence of ' android:usesCleartextTraffic ' = "false", which indicates that cleartext (unencrypted) communication is disallowed. Enabling cleartext communication can expose sensitive data to eavesdropping.

3.5.3 Debugging Enabled

Debugging functionality is crucial during the development phase but should be disabled in production applications. The scanner reviews the AndroidManifest.xml file for the presence of ' android:debuggable ' ="true", which implies that debugging is enabled. Debuggable apps may expose sensitive information and pose security risks.

3.5.4 Hardcoded PEM Files

Security best practices recommend against hardcoding private keys and certificates in mobile applications. The scanner searches for hardcoded ' .pem ' files within the decompiled APK, as their presence may indicate a security risk. Such files should be securely managed and stored.

**3.6 Framework Detection:**

In addition to identifying misconfigurations and vulnerabilities, the scanner determines the framework used to build the mobile application. Specifically, it checks for the presence of libflutter.so as an indicator of the Flutter framework. Recognizing the underlying framework aids in understanding the application's architecture and potential security implications associated with that framework.

**3.7 Insecure Mobile Device Data Storage:**

While the scanner automates the identification of vulnerabilities within the application, it also emphasizes the importance of manual verification. This section guides users and developers through the process of manually checking the device's data storage for sensitive information stored by the application. Users are encouraged to enter fake information into the application, create backups, and explore the device's storage to ensure that sensitive data is not exposed or stored insecurely.  
  
**Experimental Analysis:**

V. CONCLUSION

The "Mobile Application Vulnerability Scanner" introduced in this paper constitutes a pivotal advancement in the realm of mobile application security. In a landscape characterized by the increasing integration of mobile applications into daily life, robust security measures are imperative. The comprehensive methodology detailed herein provides a structured approach for security experts, developers, and end-users to scrutinize and enhance the security of mobile applications. Our scanner extracts essential application data, conducts deep code analysis, identifies misconfigurations, and pinpoints outdated software. Moreover, it distinguishes the underlying frameworks of these applications. By empowering users to conduct manual verifications, it combines automation with human vigilance to ensure no vulnerability is overlooked.

Mobile application security is an ongoing challenge as the threat landscape continues to evolve. This scanner offers a vital tool for preserving user data privacy and security. Nevertheless, it should complement other security practices such as penetration testing and code reviews to provide comprehensive protection. In conclusion, the "Automated Mobile Application Vulnerability Scanner" is a pivotal safeguard in the ever-changing arena of mobile application security, representing an essential step toward countering emerging threats in this dynamic domain.

1. REFERENCES

[1] Hilmi Salih Abdullah and Subhi R. M. Zeebaree .Android Mobile Applications Vulnerabilities and Prevention Methods: A Review,May 2022, Baghdad,Iraq

[2] Hua Zhang and Wang Senmiao.Vulnerability Detection on Android Apps–Inspired by Case Study on Vulnerability Related With Web Functions, IEEE,May 2020

[3] Sreenivasa Rao Basavala, Narendra Kumar and Alok Agarrwal.Mobile Applications -Vulnerability Assessment Through the Static and Dynamic Analysis, CAC2S 2013.

[4] IKRAM UL HAQ AND TAMIM AHMED KHAN.Penetration Frameworks and Development Issues in Secure Mobile Application Development: A Systematic Literature Review,June 2021,IEEE.

.

[5]