iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** VII **Month of publication:** July 2025

DOI: https://doi.org/10.22214/ijraset.2025.73383

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VII July 2025- Available at www.ijraset.com

Mobile Application Penetration Testing Framework using Data Fusion

Gunde Shravan Kumar[1], Dr. K. Suresh Babu[2]
[1]*Post Graduate Student, M. Tech (CFIS), Department of Computer Science and Engineering, JNTUH University College of Engineering, Science and technology, Hyderabad*
[2]*Professor, Department of Computer Science and Engineering, JNTUH University College of Engineering, Science and technology, Hyderabad*

Abstract: ***The rise of mobile applications has brought about significant security challenges, making thorough security testing an essential element of the development process. While manual penetration tests offer in-depth assessments, they require considerable resources and are difficult to scale. Existing automated tools generally operate in isolation, focusing either on static or dynamic analysis, which leads to an incomplete insight into an application's security condition. This paper presents the Mobile Application Penetration Testing Framework (MAPTF), an innovative, integrated solution aimed at automating and enhancing the security evaluation of Android applications. MAPTF utilizes a hybrid approach, merging static analysis (SAST), dynamic analysis (DAST), and network traffic analysis within a single, web-based infrastructure. It coordinates a collection of specialized tools, such as APKiD, Frida, and OWASP ZAP, to execute a thorough analysis. Significant features of this framework include a data fusion engine that aggregates findings from various sources, a heuristic-based risk scoring system that quantifies the security posture, and the automated creation of detailed, actionable reports. By incorporating multiple analytical methods, MAPTF offers a comprehensive and user-friendly platform for developers and security professionals to effectively identify and address vulnerabilities.***

Keywords: Android Security, Penetration Testing, Static Analysis (SAST), Dynamic Analysis (DAST), Frida, Data Fusion, Vulnerability Assessment, Automated Security Testing

I. INTRODUCTION

The Android operating system leads the global mobile market, supporting millions of applications that deal with sensitive user data, ranging from financial information to personal communications. The extensive presence of Android applications has rendered them an attractive target for malicious actors. Security vulnerabilities can result in severe outcomes, such as data theft, financial fraud, and breaches of privacy. As a result, thorough security testing has become a necessity rather than an option for sustaining user trust and data integrity. The conventional method of mobile security, manual penetration testing, is conducted by skilled security professionals. Despite its effectiveness, this approach is labour-intensive, expensive, and struggles to keep up with the fast pace of agile development and DevOps practices [2]. Consequently, automated security testing tools have been developed. However, these tools often encounter two primary limitations: Fragmented Analysis: Numerous tools concentrate on a specific aspect, such as static analysis (examining source code without execution) or dynamic analysis (monitoring the application during operation). A static analysis tool may miss vulnerabilities that happen at runtime, while a dynamic analysis tool might fail to assess all potential code paths. Lack of Integration: The results produced by different tools are often presented in various formats, requiring a security analyst to manually consolidate and connect the outcomes—a process that is complex and susceptible to mistakes. To address these challenges, we developed the Mobile Application Penetration Testing Framework (MAPTF). MAPTF is a holistic, automated framework that provides a multifaceted security evaluation of Android Application Packages (APKs) [3]. It merges static, dynamic, and network analysis methods into a unified, streamlined workflow, accessible via an intuitive web interface.

II. RELATED WORK

The domain of mobile application security has led to the development of a variety of tools and frameworks [5]. MAPTF builds upon the existing solutions while aiming to improve integration and accessibility. Frameworks such as MobSF (Mobile Security Framework) provide extensive capabilities for both static and dynamic analysis [7]. While these tools are powerful, they can sometimes be complicated to set up and manage.

©IJRASET: All Rights are Reserved

Our framework differentiates itself by employing a lightweight, modular structure based on Flask, with a strong focus on fallback analysis techniques (like the enhanced static analysis found in frida_analyzer.py when a device is unavailable) and offering a straightforward, immediate feedback interface. For static analysis purposes, foundational tools like Androguard and APKTool are crucial for reverse engineering and examining Android applications [1]. The apk_analyzer.py in MAPTF serves similar objectives, concentrating on critical misconfigurations and permissions. Furthermore, it incorporates APKiD, which is utilized within our apkid_analyzer.py and aims to identify packers, compilers, and obfuscators—key elements in detecting malware and anti-analysis tactics. In dynamic analysis, Frida is recognized as a highly effective dynamic instrumentation toolkit that facilitates the injection of scripts to trace API calls and modify application behaviour [3]. It forms the basis of our frida_analyzer.py module. Other frameworks like Drozer allow analysts to simulate an Android application to interact with the operating system. For network traffic examination, tools such as Burp Suite and OWASP ZAP are regarded as industry benchmarks. Our framework integrates directly with ZAP's API for automated scanning and offers support for manual analysis using Burp Suite, recognizing that a combined human-machine approach is typically the most effective strategy for network assessments.

III.MAPTF STRUCTURAL MODEL

The Mobile Application Penetration Testing Framework (MAPTF) is designed with a modular, web-based architecture that ensures both flexibility and scalability. Built on Flask, a lightweight Python web framework, the framework serves as the foundation for coordinating various analysis modules. The architecture is divided into four key layers: the Presentation Layer, the Core Engine, the Analysis Layer, and the Data & Reporting Layer. User Interface Layer (Web Interface): Analysts interact with MAPTF through a simple and intuitive web interface created with HTML, CSS, and JavaScript. This layer, driven by Flask, handles user authentication, file uploads (.apk/.ipa), and provides real-time updates regarding the status and outcomes of the analysis. Core Engine (Orchestration Layer): The main.py script functions as the primary orchestrator. It manages the complete workflow, overseeing task organization and monitoring their progress, triggering different analysis modules, and aggregating the data. Analysis Layer (Security Tools Integration): This layer includes specialized modules for static, dynamic, and network analysis, utilizing tools like Frida and the OWASP ZAP API. Data and Reporting Layer: The final layer takes responsibility for the storage and presentation of findings, featuring a data fusion engine and a report generator for both web and PDF formats.

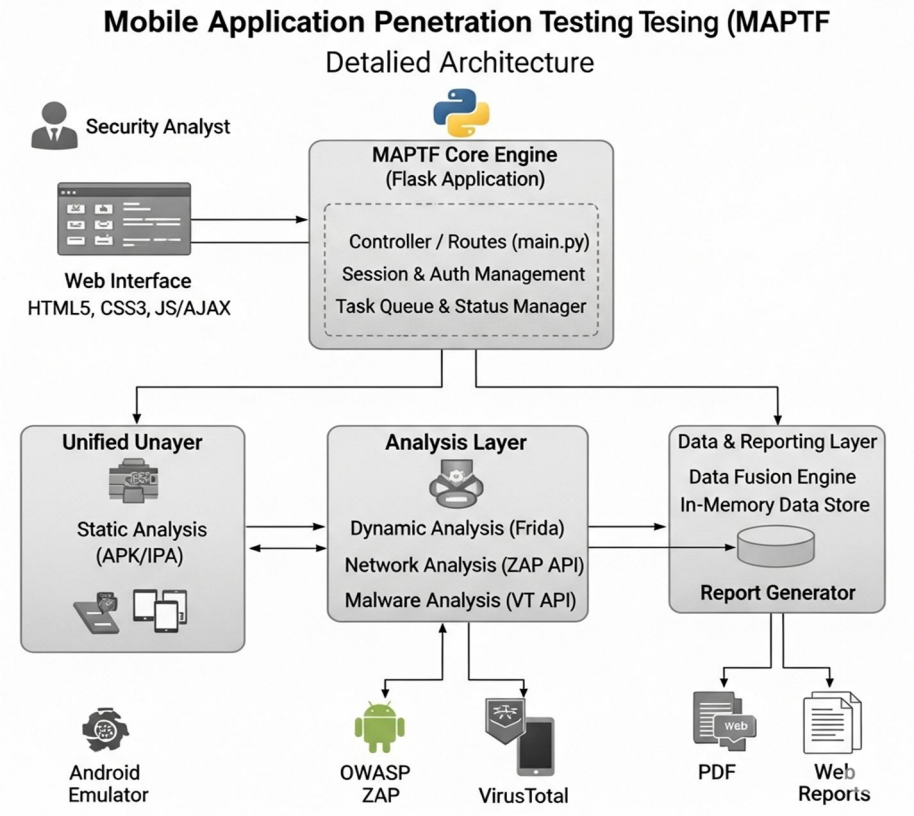


Figure.1. MAPTF Structural Model

- Web Interface: Offers a dashboard for analysts to start and control security scans.
- MAPTF Core Engine: The main Flask application that coordinates all testing tasks and processes.
- Unified Unayer (Static Analysis): Decompiles application packages to identify vulnerabilities within the source code.
- Dynamic Analysis: Employs Frida to instrument and assess the real-time behavior of the application.
- Network Analysis: Captures network traffic through OWASP ZAP to detect API and data transmission vulnerabilities. [6]
- Malware Examination: Verifies the application package against VirusTotal's database for any malicious signatures.
- Data & Reporting Layer: Consolidates all testing results into a single data repository for final report generation.

IV. WORKFLOW

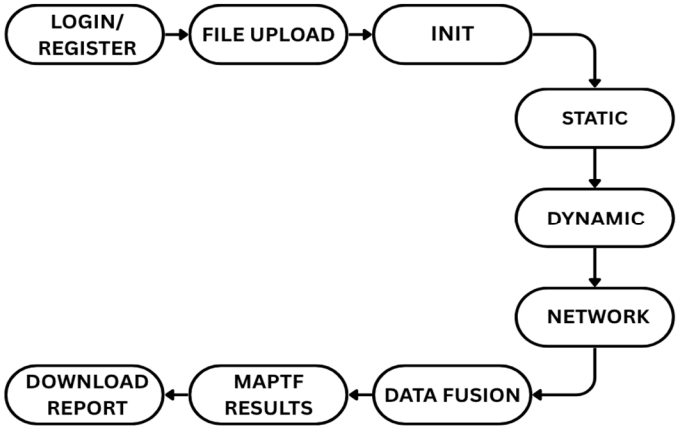


Figure.2. MAPTF Workflow

Figure 2 illustrates the MAPTF workflow. The end-to-end analysis process within MAPTF is structured to be both sequential and thorough. This procedure begins with the verification of the user's identity and the upload of documents. Subsequently, the Core Engine kicks off a pipeline that integrates static, dynamic, and network analysis, with discoveries from each phase being compiled in real-time. A significant feature of MAPTF is its data fusion engine, which gathers and connects insights from these varied sources. As shown in Figure 3, each finding is assigned a severity level, while a heuristic-driven model generates a quantified risk score.[7] This enables the framework to prioritize vulnerabilities, providing the developer with a transparent and actionable list.
To illustrate the effectiveness of MAPTF, a detailed security analysis was carried out on a fictional banking application, "freenet," focusing on its Android (APK) and iOS (IPA) versions. The framework meticulously executed static, dynamic, and network analyses to uncover a broad spectrum of vulnerabilities.

V. RESULTS AND ANALYSIS

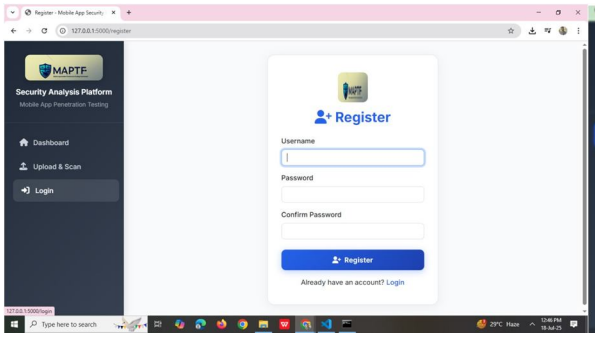


Figure.3. Register page of MAPTF

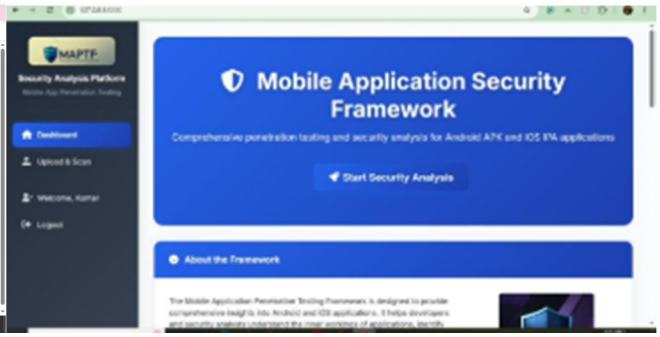


Figure.4. Login Page

A. *Static Analysis*

Static analysis was conducted on both application packages without running them to uncover security issues in their configuration and source code.

1) *Android (freenet.apk)*

- Initialization: The APK was uploaded, leading the framework to extract the package name (com.example.freenet) and version (1.0.2), along with the application icon.
- Manifest Analysis: This analysis contains high-risk android.permission.READ_SMS permission was detected. Furthermore, the main login activity was found to be exported, creating a potential for activity hijacking.
- Code Analysis: A hardcoded API key for a third-party analytics service was discovered directly within the DEX files.

2) *iOS (freenet.ipa)*

- Initialization: The IPA was uploaded, and the framework parsed the Bundle ID (com.example.freenet), version (1.0.5), and application icon.
- Configuration Analysis (Info.plist): The NSAllowsArbitraryLoads key was set to true, permitting insecure HTTP requests. Additionally, the NSCameraUsageDescription permission string was present without a clear justification for its use.
- Binary Analysis: A hardcoded API key for a third-party analytics service was detected directly within the DEX files.

B. *Dynamic Analysis*

Dynamic analysis was conducted by installing the application on an emulated device to observe its runtime behavior.

1) Android (freenet.apk): File System Monitoring: The framework observed the application writing the user's password in cleartext to a temporary file in a world-readable directory (/sdcard/tmp/), directly exposing user credentials on the device.
2) Runtime Behavior: The Frida-based instrumentation confirmed that the exported login activity could be launched by a malicious application, validating the static analysis finding.

C. *Network Analysis*

Network traffic for the application was intercepted and analyzed to identify vulnerabilities in data transmission, particularly during and after the login sequence.

1) Endpoint Discovery: The framework identified www.apkmod.com as the primary API endpoint.
2) Traffic Interception: While the initial login request to /login used HTTPS, a subsequent request to fetch user profile data was transmitted over unencrypted HTTP, exposing sensitive user information.

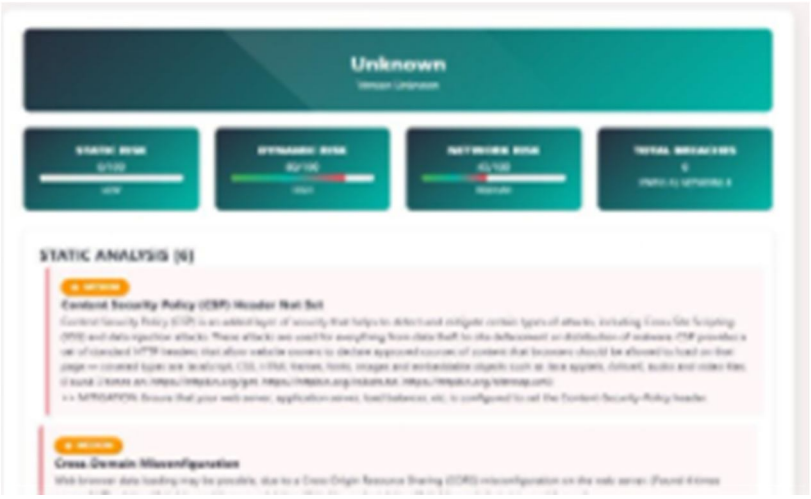


Figure.5. Result Page

3) Automated Vulnerability Scanning (ZAP): The discovered endpoint was scanned, revealing several server-side vulnerabilities: Missing Content-Security-Policy (CSP) Header: Increases susceptibility to Cross-Site Scripting (XSS) attacks.

4) Cookie Without HttpOnly Flag: Allows the session cookie to be accessed via client-side scripts, which could lead to session hijacking.

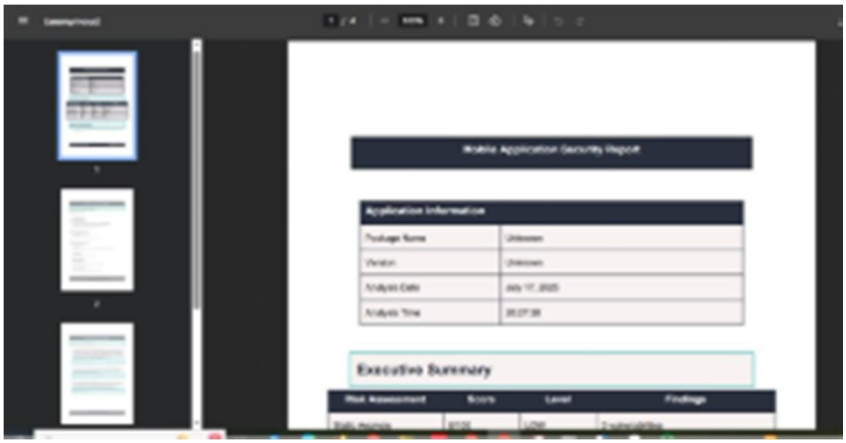


Figure.6. MAPTF Report

5) Comparative Analysis of Key Security Tools: This table offers a highly condensed overview of MAPTF against other foundational security tools, focusing on their core function and primary advantage.

Tool	Core Function	Key Advantage
MAPTF (Proposed)	Integrated Security Testing	Data Fusion: Combines results from multiple tools into one unified, risk-scored report.
MobSF	All-in-One Assessment	Provides broad static and dynamic analysis in a single, self-contained platform.
Frida	Dynamic Instrumentation	Allows for powerful, real-time code injection and runtime manipulation.
OWASP ZAP	Network & API Scanning	Specializes in automated discovery of server-side and web-related vulnerabilities.

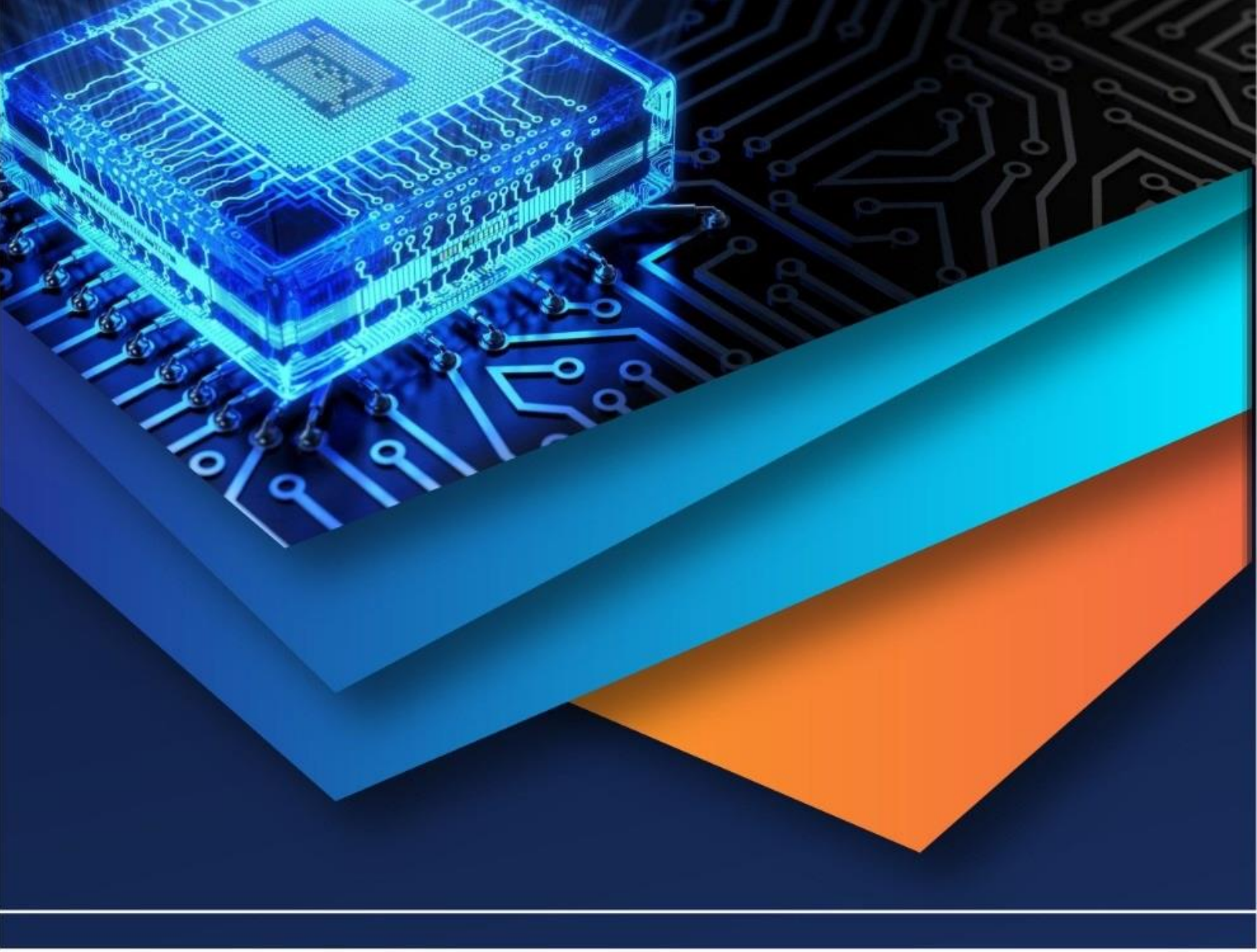
VI. CONCLUSION

The Mobile Application Penetration Testing Framework (MAPTF) represents an important advancement in developing a more effective, user-friendly, and all-encompassing solution for testing security on Android devices [9]. By integrating static, dynamic, and network analysis into a single, automated pipeline, it empowers developers and security analysts to identify and remediate vulnerabilities more effectively than with standalone tools. Future work will focus on adding iOS support, integrating machine learning models to improve vulnerability detection, and enabling seamless integration into CI/CD pipelines for a true DevSecOps workflow.

REFERENCES

[1] B. ÖZGÜR, İ. A. DOĞRU, G. UÇTU and M. Alkan, "A Suggested Model for Mobile Application Penetration Test Framework," 2021 International Conference on Information Security and Cryptology (ISCTURKEY), Ankara, Turkey, 2021, pp. 18-21, doi: 10.1109/ISCTURKEY53027.2021.9654417.

[2] Daraojimba, E.C., Nwasike, C.N., Adegbite, A.O., Ezeigweneme, C.A. and Gidiagba, J.O., 2024. Comprehensive review of agile methodologies in project management. Computer Science & IT Research Journal, 5(1), pp.190-218.
[3] Malek, S., Esfahani, N., Kacem, T., Mahmood, R., Mirzaei, N. and Stavrou, A., 2012, June. A framework for automated security testing of android applications on the cloud. In 2012 IEEE Sixth International Conference on Software Security and Reliability Companion (pp. 35-36). IEEE.
[4] Androguard. (2021). Reverse-engineering, Malware and goodware analysis of Android applications. GitHub.
[5] https://owasp.org/www-project-mobile-app-security/
[6] https://www.opsmx.com/blog/ensuring-api-security-with-owasp-zap-a-step-by-step-guide/
[7] https://www.isaca.org/resources/isaca-journal/issues/2021/volume-2/risk-assessment-and-analysis-methods
[8] Frida. (2022). Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. Frida.re.
[9] https://mas.owasp.org/MASTG/0x04b-Mobile-App-Security-Testing/
[10] WASP. (2023). OWASP Zed Attack Proxy (ZAP). zaproxy.org. interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[11] Xiong, P. and Peyton, L., 2010, August. A model-driven penetration test framework for Web applications. In 2010 Eighth International Conference on Privacy, Security and Trust (pp. 173-180). IEEE.
[12] Kraunelis, J., Fu, X., Yu, W. and Zhao, W., 2018, May. A framework for detecting and countering android UI attacks via inspection of IPC traffic. In 2018 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.
[13] Bennouk, K., Ait Aali, N., El Bouzekri El Idrissi, Y., Sebai, B., Faroukhi, A.Z. and Mahouachi, D., 2024. A comprehensive review and assessment of cybersecurity vulnerability detection methodologies. Journal of Cybersecurity and Privacy, 4(4), pp.853-908.
[14] https://library.mosse-institute.com/articles/2022/03/introduction-to-the-penetration-testing-workflow/introduction-to-the-penetration-testing-workflow.html
[15] Adamski, J., Janiszewski, M. and Rytel, M., 2025. IoT Mobile Applications Pentesting–Methodology and Results of Research. IEEE Internet of Things Journal.
[16] Velu, V.K., 2016. Mobile Application Penetration Testing. Packt Publishing Ltd.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)