IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

National Level Conference – AITCON 2K25

Adarsh Institute of Technology & Research Centre, Vita, Maharashtra

Vol. 12, Special Issue 1, March 2025

App Vulnerabilities detection

Mrs. Dipali Swapnil Lale[1], Miss. Saniya Mohammadhanif Mulani [2],

Mr. Omkar Chandrakant Mandale [3], Mr. Suraj Kundalik Kadam [4], Mr. Swapnil Sunil Kirdat[5]

Lecturer, AITP, Vita, sangli, India[1]

Lecturer, AITP, Vita, sangli, India[2]

Students, AITP, Vita, sangli, India[3]

Students, AITP, Vita, sangli, India[4]

Students, AITP, Vita, sangli, India[5]

Abstract: In this study, we analyzed common vulnerabilities and evolving attack methodologies to develop improved detection techniques. Our research focuses on two key areas: comprehensive vulnerability detection and malware infection testing strategies. Through on-site testing and detailed analysis, we identified prevalent security flaws in IoT devices and developed a suite of tools tailored for detecting these vulnerabilities.

Additionally, we discovered that some devices exhibit inherent immunity to specific malware strains, emphasizing the need for novel malware infection detection strategies. Real-world evaluations uncovered previously unknown vulnerabilities and weaknesses, revealed widespread susceptibility to DoS attacks, and demonstrated that not all devices are vulnerable to malware infections. These findings confirm the effectiveness of our approach in identifying risks and enhancing IoT security.
With the increasing prevalence of IoT devices, security vulnerabilities and malware infections have emerged as significant risks. To address these challenges, advanced vulnerability detection tools are essential for enhancing IoT security assessments.

We investigate the security of three types of Android Apps including finance, shopping and browser which are closely related to human life. And we analyze four vulnerabilities including Improper certificate validation(CWE-295:ICV), WebView bypass certificate validation vulnerability(CVE-2014-5531:WBCVV), WebView remote code execution vulnerability(CVE-2014-1939:WRCEV) and Alibaba Cloud OSS credential disclosure vulnerability(CNVD-2017-09774:ACOCDV). In order to verify the effectiveness of our analysis method in large-scale Apps on the Internet, we propose a novel scalable tool - VulArcher, which is based on heuristic method and used to discover if the above vulnerabilities exist in Apps. We download a total of 6114 of the above three types of samples in App stores, and we use VulArcher to perform the above vulnerability detection for each App. We perform manual verification by randomly selecting 100 samples of each vulnerability. We find that the accuracy rate for ACOCDV can reach 100%, the accuracy rate for WBCVV can reach 95%, and the accuracy rate for the other two vulnerabilities can reach 87%. And one of vulnerabilities detected by VulArcher has been included in China National Vulnerability Database (CNVD) ID(CNVD-2017-23282).

In addition to traditional vulnerability detection methods, modern techniques increasingly rely on machine learning (ML) and artificial intelligence (AI) to automate the identification of potential threats. These methods can analyze vast amounts of code or app behavior and recognize patterns that may indicate previously unknown vulnerabilities. Machine learning models can be trained to detect anomalous behavior, unusual patterns of system resource usage, or deviations from normal user interactions that might suggest an attack, such as privilege escalation or denial-of-service attempts.

Moreover, static application security testing (SAST) and dynamic application security testing (DAST) are widely employed in app vulnerability detection. SAST inspects an application’s source code or binary without executing the program, allowing early detection of coding flaws before deployment. DAST, on the other hand, involves testing the application while it is running, simulating how a potential attacker might exploit the application in real-world scenarios. Both methods complement each other and are crucial in identifying vulnerabilities at different stages of the app development lifecycle.

Another evolving area of vulnerability detection is the use of fuzz testing, where automated tools input random or unexpected data into an app to identify how it behaves under abnormal conditions. This method is especially useful in discovering buffer overflows, memory corruption, and other issues that are often missed by traditional testing approaches. Additionally, with the increasing complexity of modern apps that integrate third-party libraries and APIs, vulnerability scanning must also address dependencies and third-party components that could introduce risks.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

National Level Conference – AITCON 2K25

Adarsh Institute of Technology & Research Centre, Vita, Maharashtra

Vol. 12, Special Issue 1, March 2025

promptly addressed, minimizing the window of opportunity for attackers. Furthermore, the implementation of secure coding practices, regular security audits, and compliance with industry standards, such as OWASP top 10, provide a comprehensive approach to vulnerability detection and remediation.

Ultimately, proactive vulnerability detection is not only a technical requirement but also a critical element of maintaining user trust and privacy in an era where cyber threats are rapidly evolving. By adopting a multi-faceted approach that combines various detection methodologies, development teams can better safeguard their applications against emerging vulnerabilities and potential exploits.

Keywords: vulnerabilities, attack methodologies, VulArcher, dynamic application security testing

I. INTRODUCTION

In the rapidly evolving landscape of mobile and web applications, security has become a paramount concern for both developers and users. With the increasing reliance on digital platforms for communication, financial transactions, and personal data storage, vulnerabilities within these applications present significant risks to privacy, data integrity, and overall system reliability. App vulnerabilities, whether arising from coding errors, misconfigurations, or inadequate security measures, can be exploited by malicious actors, leading to severe consequences such as data breaches, unauthorized access, and service disruptions. As the complexity of applications grows, traditional methods of vulnerability detection are often insufficient to address emerging threats. This paper explores advanced techniques for app vulnerability detection, emphasizing the integration of static and dynamic analysis, machine learning, and automated security tools to proactively identify and mitigate security flaws. By providing an in-depth analysis of current methods, challenges, and solutions in vulnerability detection, this study aims to contribute to the development of more secure and resilient applications in the face of an increasingly sophisticated threat landscape.

As mobile and web applications become more integrated into everyday life, their security becomes a critical concern. The rapid pace of technological advancements, coupled with the growing sophistication of cyberattacks, presents significant challenges for developers in ensuring the integrity and safety of their applications. App vulnerabilities can manifest in various forms, from basic coding errors to complex security flaws, and can be exploited to launch attacks such as data theft, remote code execution, or denial-of-service attacks. In particular, the integration of third-party services, libraries, and APIs increases the potential for vulnerabilities, as these external components may contain undiscovered flaws or introduce new attack vectors.

The detection of vulnerabilities in applications requires a multi-faceted approach, leveraging a combination of traditional and modern security techniques to identify weaknesses before they can be exploited. Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are fundamental methods used to analyze and identify vulnerabilities in both the source code and the runtime environment. However, as applications evolve and become more complex, these methods alone may not be sufficient to identify all potential risks. Emerging techniques such as machine learning-based models and fuzz testing are gaining traction in vulnerability detection, providing more comprehensive coverage and enabling faster identification of previously unknown vulnerabilities.

In addition to identifying vulnerabilities during the development and testing phases, continuous monitoring and real-time threat detection have become increasingly vital post-deployment. With the constant release of updates, patches, and new features, applications are continuously exposed to new security risks. As a result, maintaining application security requires a proactive and ongoing effort to monitor for potential exploits and rapidly respond to any discovered vulnerabilities. This paper aims to explore the state-of-the-art methodologies for detecting app vulnerabilities, the challenges faced in this field, and potential solutions to enhance security in modern applications, ensuring that they remain resilient to evolving threats.

As the scope of app development continues to expand, security has emerged as one of the most pressing challenges in ensuring the reliability and integrity of digital platforms. The increasing sophistication of cyber threats, coupled with the growing complexity of applications, necessitates the adoption of advanced vulnerability detection methods to safeguard sensitive user data, maintain privacy, and ensure the proper functioning of systems. Vulnerabilities in apps can lead to devastating consequences, including unauthorized access to personal information, system breaches, financial losses, and damage to an organization's reputation. As a result, effective vulnerability detection has become a key component of secure app development practices.

Despite the growing awareness of security risks, traditional testing methods often fall short in addressing the new wave of vulnerabilities introduced by evolving technologies such as cloud computing, Internet of Things (IoT), and artificial intelligence (AI).

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

National Level Conference – AITCON 2K25

Adarsh Institute of Technology & Research Centre, Vita, Maharashtra

Vol. 12, Special Issue 1, March 2025

II. DETECTION REVIEW

App vulnerabilities detection is a critical part of ensuring the security and safety of any mobile, web, or desktop application. Here's a review of the main aspects of app vulnerabilities detection, methods used, and the common vulnerabilities.

LITERATURE REVIEW

Understanding Android App Security: An In-Depth Guide [here]

(https://www.getastra.com/blog/mobile/android/android-app-security/)

Security Analysis on Android Application Through Penetration Testing using Reverse Engineering [here]

(https://ieeexplore.ieee.org/document/10127653)

Predicting Android Application Security and Privacy Risk with Static Code Metrics[here]

(https://ieeexplore.ieee.org/abstract/document/7972729)

Mobile-Security-Framework-MobSF [here] (https://github.com/MobSF/Mobile-Security-Framework- MobSF)

OWASP Mobile Application Security [here] (https://owasp.org/www-project-mobile-app-security/)

Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms [here] (https://ieeexplore.ieee.org/document/8058358)

Detection Methods

Static Application Security Testing (SAST): This method analyzes the source code of an application without executing it. It scans for security issues like insecure coding practices, vulnerable libraries, and more.

Dynamic Application Security Testing (DAST): This method tests running applications (in a live environment or staging) to identify vulnerabilities in real-time, such as authentication flaws or data leakage.

Software Composition Analysis (SCA): Scans the application's dependencies and libraries to find known vulnerabilities in open-source components.

Interactive Application Security Testing (IAST): This combines both static and dynamic testing. IAST tools work within the app during runtime to pinpoint vulnerabilities.

Manual Code Review: A human expert inspects the source code to detect security flaws that automated tools might miss.

Penetration Testing: Ethical hackers simulate real-world attacks to identify weaknesses in an application's defense systems.

Tools for Vulnerability Detection

Static Analysis Tools: Examples include **SonarQube**, **Checkmarx**, and **Fortify**. These tools are used to scan the application's source code and identify issues before the app is even deployed.

Dynamic Analysis Tools: Tools like **OWASP ZAP**, **Burp Suite**, and **Acunetix** scan running applications for security issues.

Dependency Scanners: Tools like **Snyk**, **WhiteSource**, and **Black Duck** identify vulnerabilities in open-source components and libraries.

Penetration Testing Tools: Tools like **Metasploit** and **Kali Linux** contain various utilities to simulate attacks on an app and identify weaknesses.

Vulnerability Scanning Process

Discovery: Identifying the components of the app (source code, APIs, libraries, etc.) to be scanned.

Scanning: Running tools to check for vulnerabilities in the app's code, configuration, and third-party components.

Analysis: Reviewing the scan results to prioritize vulnerabilities based on risk severity.

Remediation: Fixing the identified vulnerabilities by applying patches, rewriting code, or configuring security settings properly.

Verification: Re-testing the app after fixing vulnerabilities to ensure the issues are resolved and no new vulnerabilities have been introduced.

Continuous Monitoring: Implementing ongoing security practices to continuously identify and address vulnerabilities as the app evolves.

Best Practices for Vulnerability Management

Adopt Secure Coding Practices: Always follow secure coding guidelines to prevent common vulnerabilities like SQL injection and XSS.

Use Dependency Management: Keep third-party libraries up to date and avoid using outdated versions that may contain known vulnerabilities.

Implement Strong Authentication and Authorization: Ensure robust user verification and access control mechanisms.

Encrypt Sensitive Data: Use strong encryption standards (e.g., AES) for sensitive data both at rest and in transit.

Conduct Regular Security Testing: Regularly perform SAST, DAST, and penetration tests to keep your app secure.

Educate Developers: Developers should stay updated with the latest security trends and vulnerabilities to write secure code from the beginning.

Emerging Trends in App Vulnerabilities

AI and ML in Vulnerability Detection: Artificial Intelligence (AI) and Machine Learning (ML) are being used to improve vulnerability detection by automating the identification of complex vulnerabilities and improving detection accuracy.

Shift-Left Security: The practice of integrating security into the early stages of the development lifecycle, rather than addressing vulnerabilities later in the process.

API Security: With the growing number of API-based apps, there is an increasing focus on ensuring APIs are secure and properly authenticated.

III. PROPOSED SOLUTION / METHODOLOGY

The framework is **technically feasible** using well-known **static analysis tools** (like MobSF and SonarQube) and **machine learning algorithms** (such as SVM and Random Forest). It easily integrates with **CI/CD pipelines** and can handle different application sizes, using standard **technologies** and **expertise**

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

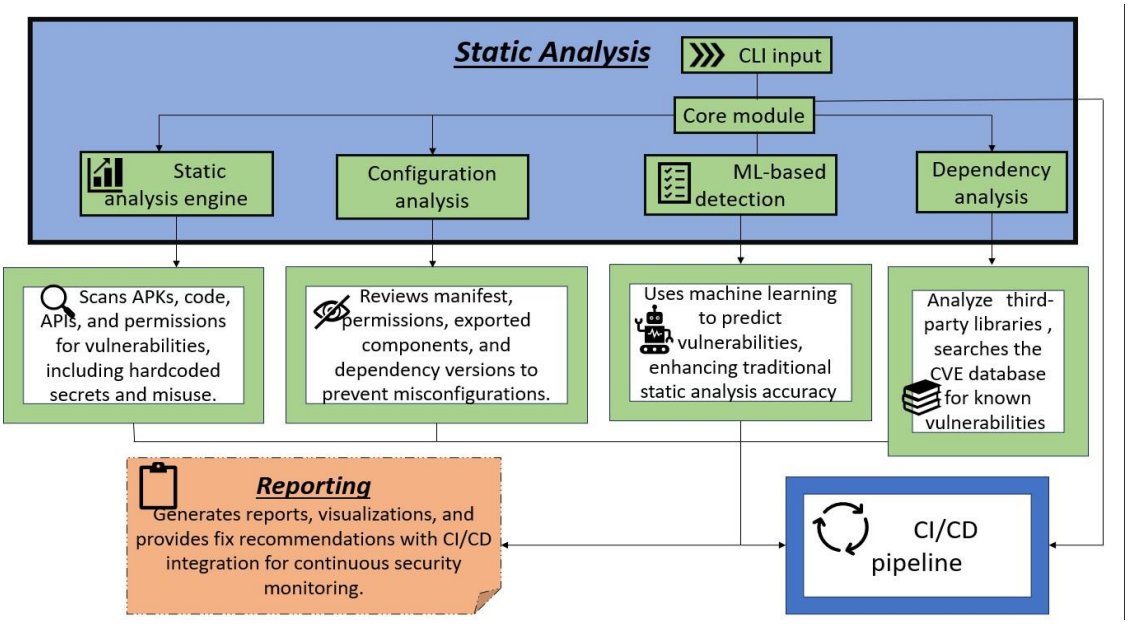
National Level Conference – AITCON 2K25

Adarsh Institute of Technology & Research Centre, Vita, Maharashtra

Vol. 12, Special Issue 1, March 2025

Key Challenges are Ensuring seamless **integration** of diverse static analysis tools and **machine learning algorithms**, while maintaining **accuracy** in vulnerability detection, managing **dependencies**, and scaling within **automated CI/CD pipelines**.

Understanding Android App Security: An In-Depth Guide [here]
(https://www.getastra.com/blog/mobile/android/android-app-security/)
Security Analysis on Android Application Through Penetration Testing using Reverse Engineering [here]
(https://ieeexplore.ieee.org/document/10127653)
Predicting Android Application Security and Privacy Risk with Static Code Metrics[here]
(https://ieeexplore.ieee.org/abstract/document/7972729)
Mobile-Security-Framework-MobSF [here] **(https://github.com/MobSF/Mobile-Security-Framework-MobSF)**
OWASP Mobile Application Security [here] **(https://owasp.org/www-project-mobile-app-security/)**
Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms [here] **(https://ieeexplore.ieee.org/document/8058358)**



TECHNOLOGIES STACK

STREAMLIT
PYTHON
MACHINE LEARNING

IV. CONCLUSION

THE PROJECT "APP VULNERABILITIES DETECTION" SERVES AS A POWERFUL TOOL FOR EARLY-STAGE VULNERABILITY DETECTION, PROVIDING AN ACCESSIBLE INTERFACE FOR DEVELOPERS TO DETECT AND FIX COMMON SECURITY ISSUES IN THEIR APPLICATIONS. WHILE THE CURRENT IMPLEMENTATION IS SIMPLE, ITS SCALABILITY AND POTENTIAL FOR FUTURE FEATURES OFFER GREAT PROMISE FOR ADVANCING APP SECURITY IN A USER-FRIENDLY, AUTOMATED MANNER.

THE PROJECT CONTRIBUTES TO THE GROWING NEED FOR AFFORDABLE, EASY-TO-USE SECURITY TOOLS IN APP DEVELOPMENT AND CAN BE A VALUABLE ASSET FOR TEAMS LOOKING TO INTEGRATE SECURITY PRACTICES EARLY IN THEIR WORKFLOWS.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

National Level Conference – AITCON 2K25

Adarsh Institute of Technology & Research Centre, Vita, Maharashtra

Vol. 12, Special Issue 1, March 2025

REFERENCES

[1]. **Snyk**: A tool that finds vulnerabilities in open-source dependencies and suggests fixes.
URL: Snyk
[2]. **WhiteSource**: A platform that helps organizations to manage open-source security vulnerabilities and license compliance.
URL: WhiteSource
[3]. **Black Duck**: Provides open-source security and license compliance management tools.
URL: Black Duck
[4]. **Burp Suite**: A leading platform for web application security testing. It includes a web vulnerability scanner and various tools for penetration testing.
URL: Burp Suite
[5]. **Acunetix**: An automated web application security scanner that detects vulnerabilities like SQL injection, XSS, and others.
[6]. **National Vulnerability Database (NVD)**: A comprehensive database of known vulnerabilities, maintained by NIST. It provides CVE (Common Vulnerabilities and Exposures) data and more.
URL: NVD
[7]. **CVE Details**: A database that provides information on CVEs and their severity ratings, helping developers prioritize vulnerabilities to fix.
URL: CVE Details
[8]. **OWASP WebGoat**: An intentionally vulnerable web application for practicing and learning about web application security.
URL: WebGoat
[9]. **PortSwigger Web Security Academy**: Free online training focused on web application security, offering tutorials on various vulnerabilities.
URL: Web Security Academy
[10]. **Cybrary**: Offers various cybersecurity courses, including on application security and penetration testing.
URL: Cybrary
[11]. **NIST Cybersecurity Framework**: A set of guidelines and best practices to help organizations manage cybersecurity risk, including software vulnerabilities.
URL: NIST Cybersecurity Framework
[12]. **CIS Controls**: A set of best practices for securing IT systems and data, including guidelines for securing applications.
URL: CIS Controls
[13]. **ISO/IEC 27001**: International standards for managing information security, including the management of application security vulnerabilities.
URL: ISO/IEC 27001