A Comprehensive Automated Security Testing Tool for Flutter Applications

Dr. V. Kavitha,
Associate Professor, Department of Computer Science Engineering,
Velalar College of Engineering and Technology,
Erode, Tamil Nadu, India.
Shyam Kishore V, Sreeraam K, Gopalakrishnan G
Student, Department of Computer Science Engineering,
Velalar College of Engineering and Technology,
Erode, Tamil Nadu, India.

Abstract: The increasing use of mobile devices for everyday tasks has made mobile computing a primary means of storing sensitive and private data. Mobile application stores act as intermediaries between users' devices and third-party developers, making it crucial for these stores to ensure high security and quality standards for the apps available for download. However, the growing popularity of Flutter apps has led to an increase in security vulnerabilities, which can potentially expose users' sensitive data. To address this issue, we developed an automated security testing tool for Flutter applications. Our tool uses dynamic, static, network and app-layer analysis techniques to detect vulnerabilities in real-time and generate detailed reports with recommendations for fixing them. Our testing on real-world Flutter applications demonstrated the tool's effectiveness in detecting a wide range of security vulnerabilities. The tool is designed to be user-friendly and can help Flutter developers improve the security of their applications and safeguard users' sensitive data.

Keywords— mobile devices, vulnerabilities, feedback, data exposure risk, flutter applications, security recommendations, developers

I. INTRODUCTION

Mobile computing is becoming increasingly popular, with people relying more and more on their mobile devices to perform even the most basic tasks and store their most sensitive data. With mobile application stores serving as the gatekeepers between third-party app developers and users' mobile devices, it's essential that these stores ensure that apps meet high security and quality standards to minimize the risk of exposing users' data. One crucial way to achieve these standards is through early testing and detection of vulnerabilities in mobile applications, which can result from poor development practices. By providing developers with feedback on these issues and guidance on how to address them, app stores can help to improve the overall security and quality of mobile apps.

Mobile applications are often vulnerable to various security threats such as unauthorized access, data leakage, and malware attacks. These threats can result in significant losses of sensitive data and can even result in reputational damage or financial loss for the users of the applications. As a result, it is essential to incorporate robust security testing measures during the development process to detect and address vulnerabilities before they can be exploited by attackers.

Flutter is a popular mobile application development framework that has gained significant attention due to its expressive, high-quality, and flexible user interface capabilities. However, as the use of Flutter applications continues to increase, so does the need to address the potential security vulnerabilities that can arise within them. These vulnerabilities can pose significant risks to users' sensitive data, making it essential to incorporate robust security testing measures during the development process.

Manual security testing is time-consuming and can be error-prone, which has led to an increasing interest in automated security testing for mobile applications. In this paper, we present a comprehensive automated security testing tool for Flutter applications that uses a combination of dynamic and static analysis techniques to detect vulnerabilities in real-time and provide developers with detailed reports on how to address them.

In this paper, we will discuss the various types of security testing techniques available for mobile applications, including manual and automated methods. We will also explore the advantages and limitations of each approach, as well as the challenges that developers face in implementing effective security testing processes. We will provide an overview of the most popular automated security testing tools for mobile applications and compare their features and capabilities. Finally, we will propose a comprehensive framework for automated security testing in mobile applications, which incorporates the latest techniques and methodologies for identifying and addressing security vulnerabilities. Our proposed framework will help developers to improve the security of their applications and reduce the risks of exposing users' sensitive data.

Our automated security testing tool for Flutter applications offers a valuable resource for developers seeking to improve the security of their applications and reduce the risks of exposing users' sensitive data. By providing a comprehensive and user-friendly tool for automated security testing, we hope to contribute to the advancement of mobile application security research and promote the development of more secure Flutter applications.

II. SECURITY TESTING IN MOBILE APPLICATIONS

Security testing is a critical aspect of mobile application development that involves the identification and assessment of vulnerabilities that can pose a threat to the confidentiality, integrity, and availability of users' data. Mobile applications are susceptible to various security threats such as unauthorized access, data leakage, and malware attacks. These threats can result in significant losses of sensitive data and can even result in reputational damage or financial loss for the users of the applications. As a result, it is essential to incorporate robust security testing measures during the development process to detect and address vulnerabilities before they can be exploited by attackers.

There are various types of security testing techniques available for mobile applications, including manual and automated methods. Manual security testing involves the use of human testers to identify vulnerabilities in an application by simulating real-world attack scenarios. Manual testing can be time-consuming and can be error-prone, as testers may miss critical vulnerabilities. Moreover, manual testing is not scalable, as it can become increasingly difficult to identify vulnerabilities as the size and complexity of the application increase.

On the other hand, automated security testing involves the use of software tools to detect vulnerabilities in an application automatically [9]. Automated testing is faster and more efficient than manual testing, and it can identify a broad range of vulnerabilities in real-time [1]. Automated testing can also be more accurate than manual testing, as it can identify vulnerabilities that may be missed by human testers.

There are several advantages and limitations to both manual and automated security testing methods. Manual testing is more effective in identifying complex vulnerabilities that may not be detected by automated tools [7]. However, manual testing is time-consuming and can be expensive, especially for large and complex applications. Automated testing, on the other hand, is more efficient and cost-effective, but it may not be able to detect complex vulnerabilities that require a human understanding of the application's behavior.

To ensure the security of mobile applications, it is essential to use a combination of both manual and automated security testing methods. By using a combination of both methods, developers can identify and address a broad range of vulnerabilities in their applications. Moreover, developers can use automated testing tools to detect and identify vulnerabilities in real-time [2], while manual testing can be used to validate the findings of automated tools and identify complex vulnerabilities [3] that may be missed by automated tools.

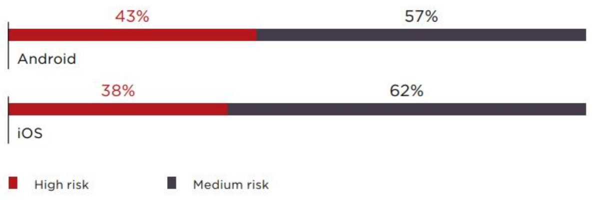


Figure 1. *Maximum risk level of vulnerabilities*

III. RELATED WORKS

In the following section of this paper, we will address the importance of security in mobile application development. Developers have a crucial role in ensuring the security of the application during its execution on end-users' devices. Malicious applications or users can target these devices to compromise user data or carry out harmful activities [6]. Therefore, developers must be aware of the various security risks that affect mobile applications and implement appropriate measures to mitigate those risks. They should also understand the actions they can take to enhance security and protect users' data [10].

A. Mobile applications development security risks

As referred before, the development of a mobile application can be affected by multiple risks. One of the primary sources for identifying and classifying risks in mobile application development is OWASP. OWASP's Mobile Security Project includes several projects related to mobile application security . The OWASP Top 10 Mobile Risks is a list that summarizes the ten most prevalent risks affecting the security of mobile applications, including (M1) Improper Pplatform Usage, (M2) Insecure Data Sstorage, (M3) Insecure Ccommunication, (M4) In secure Authentication, (M5) Iinsufficient Ccryptography, (M6) Insecure Aauthorization, (M7) Client Code Qquality, (M8) Code Tampering[2], (M9) Rreveres Eengineering, and (M10)Eextraneous Efunctionality.

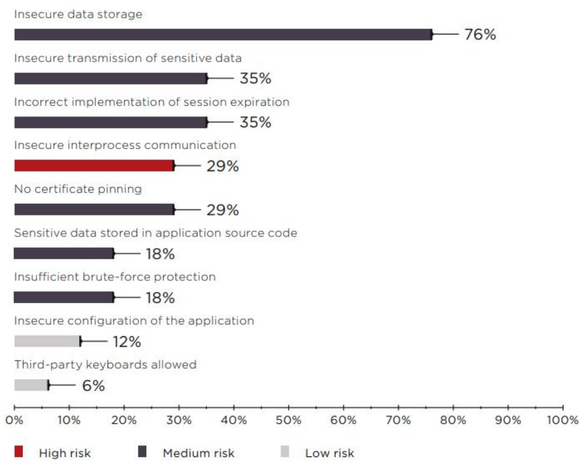


Figure 2. *Mobile application vulnerabilities*

In addition to the OWASP Mobile Top 10, which lists the most common and prevalent mobile application security risks, OWASP also provides two other relevant sources of information for mobile application security development. The first is the Mobile Security Testing Guide [8], which is a comprehensive manual for mobile application security testing and reverse engineering devoted to the ions and Android mobile platforms. The second is the OWASP Mobile Application Security Verification Standard, used by software architects and developers seeking to develop secure mobile applications[5]. This standard helps ensure the completeness and consistency of the security test results.

In the previously mentioned section, it can be deduced that there are various initiatives dedicated to the development of mobile application security. ENISA has produced the "Privacy and data protection in mobile applications" report, which is a meta-study on privacy and data protection in mobile apps, examining the app development environment's features that affect privacy and security, and identifying relevant best practices, open issues, and gaps in the field.

ENISA has also developed the "Smartphone Secure Development Guidelines" report, which provides guidance for developers of smartphone applications on developing secure mobile applications and covers aspects such as identifying and protecting sensitive data, user authentication and authorization, and consent and privacy protection.

NIST also has work in the mobile applications security field, with the most visible being the "Vetting the Security of Mobile Applications" report , which aims to help organizations understand the process for vetting mobile application security and develop application security requirements. In conclusion, there are various initiatives with different security requirements that need to be addressed by stakeholders to ensure secure mobile applications for users and organizations.

Overall, automated security testing tools offer developers a powerful way to identify potential security vulnerabilities in their mobile applications quickly. While each tool has its strengths and weaknesses, incorporating automated security testing into the mobile application development process can significantly improve the security and quality of mobile applications.

IV. PROPOSED METHOD

The proposed tool will utilize a combination of security testing tools, such as those mentioned in the previous section, to provide a comprehensive approach to automated security testing for Flutter applications [4]. The tool will provide a user-friendly GUI that can be easily navigated, reducing the need for manual intervention and technical expertise. This will make it easier for organizations with limited resources or technical knowledge to perform automated security testing and identify vulnerabilities in their applications. The tool will generate a vulnerability report that summarizes the results of the security tests performed on the Flutter application. The report will highlight any potential vulnerabilities or security issues, providing actionable insights for developers to fix the identified issues. The proposed system will be scalable and customizable, allowing organizations to tailor their security testing approach based on their specific needs and requirements. The system will provide continuous security testing and monitoring, enabling organizations to keep their Flutter applications secure and free from vulnerabilities over time.

By providing a more secure mobile ecosystem, the proposed tool will help organizations protect their users' sensitive information, build trust with their users, and avoid costly security breaches and data leaks. The tool will be developed using best practices in software development and security testing to ensure its effectiveness, reliability, and usability.

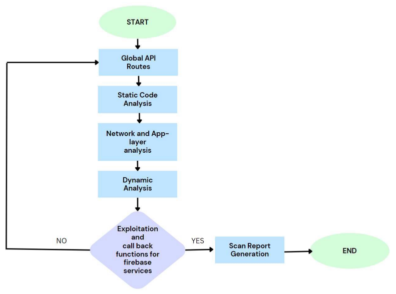
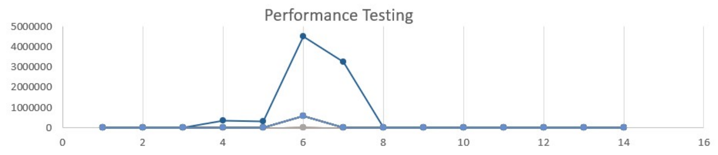

START
Global API Routes
Static Code Analysis
Network and App-layer analysis
Dynamic Analysis
Exploitation and call back functions for firebase services
NO
YES
Scan Report Generation
END

Figure 3. *Vulnerability analysis system flow*

V PERFORMANCE EVALUATION

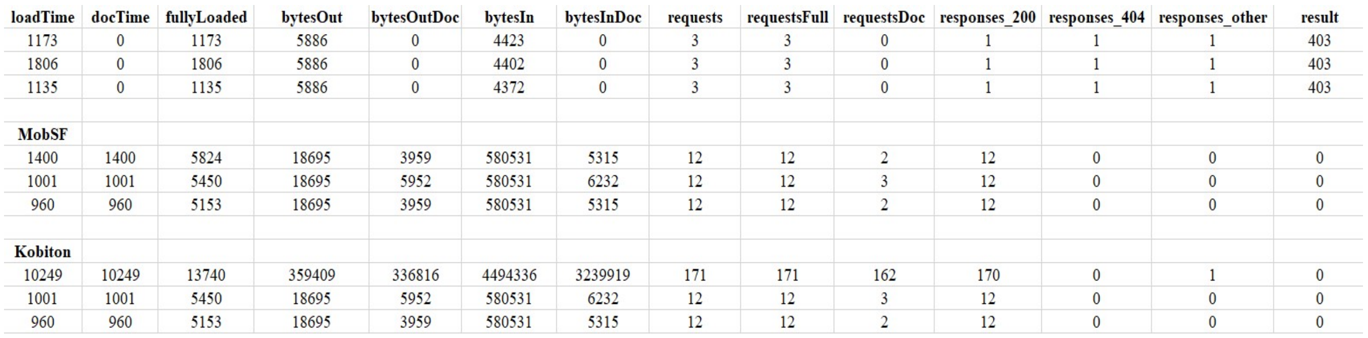
Evaluating the product is the most important part of development. It works on constructive feedback; this process is continued until a desirable accuracy is achieved. The performance of the product is measured by evaluating it. It is vital to check the accuracy of the results computed by the product. We have analyzed the performance of our product with the existing flutter apk security testing tools like Mobs and Kobiton.

The parameters are Load Time: This metric measures the amount of time it takes for the website to fully load in the user's browser. Time to First Byte: This metric measures the amount of time it takes for the server to respond with the first byte of data after a request is made. Number of Requests: This metric counts the number of HTTP requests made by the website to load all of its assets, including images, CSS files, and scripts. Doc Time: The time it takes for the web page to be fully loaded and ready to use, including all the images, scripts, and other resources that the page requires. Fully Loaded: The time it takes for the entire web page, including all resources, to be fully loaded and ready for use. Bytes Out: The amount of data that was sent from the server to the client in response to the request for the web page. Bytes out Doc: The amount of data that was sent from the server to the client in response to the request for the web page, excluding any resources that are requested after the page is loaded. Bytes In: The amount of data that was sent from the client to the server, such as form submissions or other user input. Requests: The number of requests that were made to the server to load the web page and its resources. Result: The outcome of the request, typically represented as a status code, such as 200 for a successful request or 404 for a page not found error. These parameters are useful for evaluating the performance and efficiency of a web page or website, as well as identifying potential issues and areas for improvement.



VI CONCLUSION

In conclusion, the development of a comprehensive automated security testing tool for Flutter applications is a significant step towards ensuring the security and reliability of mobile applications. By integrating security testing into the development cycle, developers can identify and address security vulnerabilities early on, reducing the risk of security breaches and improving the overall quality of the application. Moving forward, there is still much work to be done in the field of automated security testing for mobile applications. As the mobile ecosystem continues to evolve, developers will need to adapt and innovate new testing methodologies and techniques to keep pace with the changing threat landscape.



loadTime	docTime	fullyLoaded	bytesOut	bytesOutDoc	bytesIn	bytesInDoc	requests	requestsFull	requestsDoc	responses_200	responses_404	responses_other	result
1173	0	1173	5886	0	4423	0	3	3	0	1	1	1	403
1806	0	1806	5886	0	4402	0	3	3	0	1	1	1	403
1135	0	1135	5886	0	4372	0	3	3	0	1	1	1	403
MobSF													
1400	1400	5824	18695	3959	580531	5315	12	12	2	12	0	0	0
1001	1001	5450	18695	5952	580531	6232	12	12	3	12	0	0	0
960	960	5153	18695	3959	580531	5315	12	12	2	12	0	0	0
Kobiton													
10249	10249	13740	359409	336816	4494336	3239919	171	171	162	170	0	1	0
1001	1001	5450	18695	5952	580531	6232	12	12	3	12	0	0	0
960	960	5153	18695	3959	580531	5315	12	12	2	12	0	0	0

Table 1. *Performance analysis table*

VII FUTURE WORKS

- Integration with more advance testing techniques such as fuzz testing and symbolic execution to further enhance the tool's capabilities.
- Incorporating machine learning algorithms to improve the accuracy and efficiency of the testing process, as well as to predict and prevent potential security breaches.
- Enhancing the tool's compatibility with a wider range of mobile application development frameworks and technologies.
- Continuing research on emerging threats and vulnerabilities in the mobile ecosystem and updating the tool to stay current with the latest security trends and best practices.

REFRENCES

[1] E. Park et al., "Development of Automatic Evaluation Tool for Mobile Accessibility for Android Application," 2019 International Conference on Systems of Collaboration Big Data, Internet of Things & Security (SysCoBIoTS), Casablanca, Morocco, 2019, pp. 1-6, doi: 10.1109/SysCoBIoTS48768.2019.9028034.

[2] C.Nagarajan and M.Madheswaran - ‘Experimental verification and stability state space analysis of CLL-T Series Parallel Resonant Converter’ - Journal of ELECTRICAL ENGINEERING, Vol.63 (6), pp.365-372, Dec.2012.

[3] G.Neelakrishnan, K.Anandhakumar, A.Prathap, S.Prakash “Performance Estimation of cascaded h-bridge MLI for HEV using SVPWM”SurajPunj Journal for Multidisciplinary Research, 2021, Volume 11, Issue 4, pp:750-756

[4] Y. Zhauniarovich, A. Philippov, O. Gadyatskaya, B. Crispo and F. Massacci, "Towards Black Box Testing of Android Apps," 2015 10th International Conference on Availability, Reliability and Security, Toulouse, France, 2015, pp. 501-510, doi: 10.1109/ARES.2015.70.

[5] G.Neelakrishnan, S.N.Pruthika, P.T.Shalini, S.Soniya, “Perfromance Investigation of T-Source Inverter fed with Solar Cell” SurajPunj Journal for Multidisciplinary Research, 2021, Volume 11, Issue 4, pp:744-749

[6] C.Nagarajan and M.Madheswaran - ‘Performance Analysis of LCL-T Resonant Converter with Fuzzy/PID Using State Space Analysis’- Springer, Electrical Engineering, Vol.93 (3), pp.167-178, September 2011.

[7] D. Lai and J. Rubin, "Goal-Driven Exploration for Android Applications," 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 2019, pp. 115-127, doi: 10.1109/ASE.2019.00021.

[8] C.Nagarajan and M.Madheswaran - ‘Stability Analysis of Series Parallel Resonant Converter with Fuzzy Logic Controller Using State Space Techniques’- Taylor & Francis, Electric Power Components and Systems, Vol.39 (8), pp.780-793, May 2011.

[9] G.Neelakrishnan, P.Iraianbu, T.Abishek, G.Rajesh, S.Vignesh, “IOT Based Monitoring in Agricultural” International Journal of Innovative Research in Science, Engineering and Technology, March 2020, Volume 9, Issue 3, pp:814-819

[10] H. Ge, L. Ting, D. Hang, Y. Hewei and Z. Miao, "Malicious Code Detection for Android Using Instruction Signatures," 2014 IEEE 8th International Symposium on Service Oriented System Engineering, Oxford, UK, 2014, pp. 332-337, doi: 10.1109/SOSE.2014.48.

[11] Nagarajan and M.Madheswaran - ‘Experimental Study and steady state stability analysis of CLL-T Series Parallel Resonant Converter with Fuzzy controller using State Space Analysis’- Iranian Journal of Electrical & Electronic Engineering, Vol.8 (3), pp.259-267, September 2012

[12] C. Nagarajan, G.Neelakrishnan, R. Janani, S.Maithili, G. Ramya “Investigation on Fault Analysis for Power Transformers Using Adaptive Differential Relay”Asian Journal of Electrical Science, Vol.11 No.1, pp: 1-8, 2022.

[13] A. Shabtai, Y. Fledel and Y. Elovici, "Automated Static Code Analysis for Classifying Android Applications Using Machine Learning," 2010 International Conference on Computational Intelligence and Security, Nanning, China, 2010, pp. 329-333, doi: 10.1109/CIS.2010.77.

[14] Nagarajan C., Neelakrishnan G., Akila P., Fathima U., Sneha S. “Performance Analysis and Implementation of 89C51 Controller Based Solar Tracking System with Boost Converter” Journal of VLSI Design Tools & Technology. 2022; 12(2): 34–41p.

[15] B. Mathis, V. Avdiienko, E. O. Soremekun, M. Böhme and A. Zeller, "Detecting information flow by mutating input data," 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA, 2017, pp. 263-273, doi: 10.1109/ASE.2017.8115639.

[16] M. M. Saudi, F. Ridzuan, N. Basir, N. F. Nabila, S. A. Pitchay and I. N. Ahmad, "Android Mobile Malware Surveillance Exploitation via Call Logs: Proof of Concept," 2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim), Cambridge, UK, 2015, pp. 176-181, doi: 10.1109/UKSim.2015.89.