International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 7, Issue 11, pp 57-62, November-2023
https://doi.org/10.47001/IRJIET/2023.711009

Autonomous CBSL Mobile Security Compliance Testing Tool

[1]R M S A Karunadasa, [2]Y G A S Karunarathna, [3]I A R R Illankoon, [4]U G R M Dias, [5]Kanishka Yapa, [6]Amila Senarathne

[1,2,3,4,5,6]Department of Computer Systems Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka
Authors E-mail: [1]shanuka.ashen.10@gmail.com, [2]avishka.kings29@gmail.com, [3]illankoon.roshini@gmail.com, [4]ramindudias@gmail.com, [5]kanishka.y@sliit.lk, [6]amila.n@sliit.lk

***Abstract* - In response to the Central Bank of Sri Lanka's (CBSL) regulations governing mobile payment applications across various scenarios, this research introduces "MOBY GUARD," an Autonomous CBSL Mobile Security Compliance Solution. CBSL mandates compliance for new mobile payment services, service modifications, security breaches, CBSL-initiated assessments, payment gateway provider assessments, and general regulatory alignment. Traditionally, mobile application vendors developed their applications and sought external organizations to assess compliance with CBSL guidelines, permitting publication on platforms like the Google Play Store. However, MOBY GUARD presents an innovative approach, enabling payment-related mobile application owners to autonomously perform CBSL security compliance assessments. This eliminates third-party involvement, reducing time and cost overheads. By enabling in-house security checks, MOBY GUARD enhances efficiency, allowing for more frequent and thorough security evaluations. Focusing on Android applications for the Google Play Store, the project addresses 86 CBSL compliance requirements, prioritizing the three most critical ones. These components encompass root detection, code integrity checks, SSL pinning, and Smali code modifications. This approach proactively strengthens the security of payment-related mobile applications while optimizing the compliance process.**

Keywords: CBSL Regulations, Mobile Payment Applications, Security Compliance, Android Platform, Security Enhancement, Compliance Testing.

I. INTRODUCTION

The Central Bank of Sri Lanka (CBSL) has established regulations mandating mobile payment applications to adhere to a defined set of requirements across various scenarios:[1]

1) Prior to Launching a New Mobile Payment Service in Sri Lanka: Mobile payment service providers must ensure compliance with CBSL stipulations before introducing a new service.
2) When Modifying an Existing Mobile Payment Service: Changes made to an existing mobile payment service necessitate adherence to the prescribed requirements.[1]
3) In Case of Suspected Security Breach in a Mobile Payment Service: Should a security breach be suspected, immediate action must be taken to meet CBSL requirements.[1]
4) Upon Request for Assessment by the CBSL: The CBSL can initiate an evaluation, prompting mobile payment providers to undergo the assessment process.[1]
5) In Response to Assessment Requests from Payment Gateway Providers: Payment Gateway providers can ask for a CBSL assessment, compelling compliance from mobile payment services.[1]
6) When Ensuring Compliance with Applicable Regulations: Mobile payment providers need to guarantee alignment with all relevant regulations to operate effectively.[1]

Traditionally, the process involved mobile application vendors developing their applications either independently or with software solution providers. Following development, an external organization conducted security compliance testing to ascertain CBSL guideline adherence. Only upon successfully passing these guidelines could the application be published on platforms like the Google Play Store [1].

However, the proposed security solution introduces an innovative approach empowering payment-related mobile application owners to independently perform CBSL security compliance assessments. This eliminates the necessity of third-party involvement, thus reducing time and costs associated with compliance testing. By enabling in-house security checks, this solution enhances the efficiency of compliance testing, enabling more frequent and thorough evaluations of application security. Ultimately, this approach fosters a proactive CBSL compliance stance, bolstering the security of payment-related mobile applications overall.

Focusing solely on the Android platform and applications destined for the Google Play Store, this project addresses the 86 key compliance requirements within the CBSL's

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 57-62, November-2023

https://doi.org/10.47001/IRJIET/2023.711009

Compliance list.[1] While all requirements hold importance, the project will prioritize the three most critical ones, estimated to take approximately five working days to fulfill. With available resources and local personnel, the projected cost of this endeavor stands at a minimum of half a million Sri Lankan rupees.

II. LITERATURE REVIEW

The realm of mobile payment applications has witnessed substantial growth and innovation, accompanied by an increasing need for robust security measures to safeguard user data and financial transactions. In response to this evolving landscape, the Central Bank of Sri Lanka (CBSL) has instituted a series of regulations mandating compliance for mobile payment applications across various operational scenarios. This literature review aims to explore existing research and practices in the domain of mobile payment application security, with a specific focus on the implementation of the critical compliance standards outlined by the CBSL.

The introduction of CBSL's regulations represents a significant step towards fortifying the security of mobile payment applications. Traditional practices involved mobile application vendors developing their solutions either independently or with the assistance of software solution providers. Subsequently, third-party organizations conducted security compliance testing to ensure adherence to CBSL guidelines, allowing applications to be published on platforms such as the Google Play Store. However, this approach often incurred additional time and costs.

The proposed security solution, as outlined in the methodology, introduces a transformative approach by empowering payment-related mobile application owners to autonomously perform CBSL security compliance assessments. This innovative shift eliminates the reliance on third-party organizations for compliance testing and, in turn, streamlines the process. The literature surrounding mobile application security underscores the importance of a proactive stance,advocating for strategies that enable real-time assessment and adaptation to emerging threats. This approach aligns with the proposed solution's objective of conducting in-house security compliance checks, thus enhancing overall efficiency and agility in addressing security concerns.

The three critical compliance standards identified for implementation reflect pertinent challenges in the mobile application security landscape. "Root Device Detection" tackles the vulnerability associated with rooted devices, a common target for malicious attacks. Research in mobile security emphasizes the significance of addressing root detection as part of a comprehensive defense strategy against potential breaches. The proposed method of Java file analysis for root detection aligns with established practices while introducing an innovative checklist-based approach.

"Code Integrity Check" and "SSL Pinning check and bypass" address code manipulation and data interception concerns. These components resonate with contemporary security research, which highlights the significance of code integrity to prevent unauthorized code modifications and SSL pinning to secure data transmission. The methodology's recommendation-focused approach to bolstering code integrity and SSL pinning aligns with best practices for maintaining application security in the face of evolving attack vectors.

Moreover, the integration of all outputs from the individual components into a consolidated "Patched version" reflects a holistic approach to security enhancement. Research underscores the importance of comprehensive security strategies that encompass multiple layers of defence. By consolidating insights gained from individual assessments, the methodology aligns with the broader perspective of layered security measures.

In conclusion, the literature review highlights the significance of CBSL's initiative in the context of mobile payment application security. By combining established practices with innovative approaches, the proposed solution resonates with the broader landscape of mobile application security research. The proactive stance, emphasis on in-house assessments, and incorporation of multiple security components reflect a comprehensive strategy aligned with contemporary best practices. As the mobile payment landscape continues to evolve, strategies that foster agility, autonomy, and robust security are poised to play a pivotal role in ensuring the safety of financial transactions and user data.

METHODOLOGY

The following are the three critical compliance standards identified in the "Guidelines on Minimum Compliance Standards for payment related mobile Applications" published by CBSL that we have chosen to implement for this project.

Individual Component I - Root Device Detection

- CBSL guideline: 14.2 Payment related mobile applications shall not be allowed to be executed on rooted devices.[1]
- The first component of the process involves Root Implementation Detection and Magisk Hide Bypass.[2] It takes an Android application as input and proceeds by converting the APK file into a Java file for analysis. The main objective is to determine whether the application has implemented root detection mechanisms. If such

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 57-62, November-2023

https://doi.org/10.47001/IRJIET/2023.711009

mechanisms are present, the system evaluates their effectiveness.[3] Additionally, the component assesses the application's capability to detect Magisk Hide, a popular root hiding technique. If the app lacks Magisk Hide detection, the component provides recommendations for integrating this functionality. [4]As an outcome, a comprehensive checklist detailing the root and Magisk Hide detection status is generated, accompanied by suggestions to enhance Magisk Hide detection methods. This component plays a crucial role in enhancing the security and robustness of Android applications against potential security breaches related to rooted devices and root hiding mechanisms.

Individual Component II - Code Integrity Check

- CBSL guideline: 14.1 The following checks shall be implemented in the server-side to verify the integrity and to detect any manipulation of the client application. [1]These checks can be executed at the start of the payment related mobile application or as appropriate. If any of these checks fail, payment related mobile application shall be disabled.
  - Hash values/checksums of code blocks, classes, or the whole program
  - Validate the size of certain system files or the file modification timestamps.
  - Verify the signature of the package file at the run time.
- The second component focuses on Code Integrity Security Detection and Prevention within the context of Android applications.[5] Beginning with an Android application as input, the component initiates the process by converting the provided APK file into a Java file for in-depth analysis.[6] The primary objective is to assess whether the application has integrated code integrity detection measures. Regardless of the presence of such measures, the component proceeds to offer recommendations for the adoption of advanced code integrity implementation techniques. The resulting output from this component is a set of tailored recommendations aimed at bolstering the application's code integrity mechanisms. This facet of the process significantly contributes to the enhancement of the application's overall security posture by ensuring the integrity and authenticity of its code base, thereby minimizing the potential for malicious code manipulation or unauthorized modifications.

Individual Component III - SSL Pinning check and bypass

- CBSL guideline: 12.4 Controls to mitigate bypassing of certificate pinning shall be implemented.[1]
- The third component revolves around SSL Certificate Pinning Bypassing within the realm of Android applications.[7] Commencing with an Android application as the initial input, the component undertakes the conversion of the provided APK file into a Java file, subsequently subjecting it to a comprehensive evaluation. The core focus is placed on ascertaining the existence of SSL Pinning implementation within the application. In instances where SSL Pinning measures are found to be lacking, the component takes the initiative to offer tailored recommendations for the incorporation of SSL Pinning. [8]As a resultant output, the component furnishes a set of strategic suggestions designed to guide the application's integration of SSL Pinning techniques.[9] This facet of the process assumes a pivotal role in fortifying the application's security framework by advocating for the adoption of SSL Pinning, thereby fostering a robust defence against potential vulnerabilities associated with unauthorized interception or manipulation of data transmitted over the network.[10]

Regarding the final component, we will create a patch .apk file by consolidating all the outputs obtained from the aforementioned individual components.

Individual Component IV – Develop the Patched version of the initially provided .apk file using the Smali code according to root, SSL Pinning and Integrity recommendation. CBSL 12.3 Guideline will be checked.[1]

- The fourth component is centred around the Implementation of Automated Security for Android Digital Finance Applications. Its input is derived from the outputs of the preceding components, encompassing information concerning root detection, SSL Pinning, and code integrity measures. The process unfolds by establishing a linkage between the Java code and the Smali code, followed by an intricate analysis to pinpoint the precise path and function necessitating alteration. Subsequently, targeted modifications are executed on the Smali code as required, culminating in the reconstruction of the Android application. The ultimate result achieved by this component is a patched Android file, one that reflects the amalgamation of refined security enhancements obtained from the collective insights provided by the earlier stages. This component is instrumental in realizing a fortified security infrastructure tailored specifically for Android-based financial applications, ensuring a resilient shield against potential threats, manipulation, and unauthorized access, thus fostering the safe and secure utilization of digital finance services.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 57-62, November-2023

https://doi.org/10.47001/IRJIET/2023.711009

IV. RESULTS AND DISCUSSION

In this research, we present four distinct areas of focus in the field of Android application security. Each of these research topics delves into critical aspects of securing Android apps against various threats and vulnerabilities. We provide an overview of the research focus, highlighting key findings and contributions for each area. These insights collectively contribute to a better understanding of the evolving landscape of Android app security and the measures required to protect against potential risks.

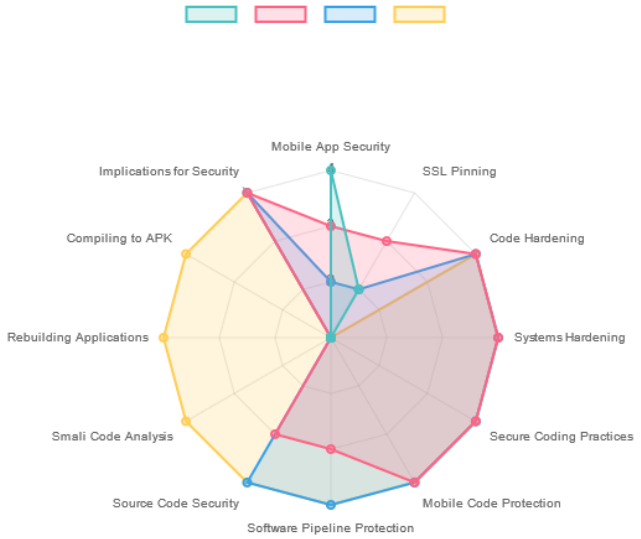

Figure 1: Research Focus Radar Chart

Table 1: Research focus and key findings and contributions

Research focus	*Key Findings and Contributions*
Detection of Magisk Hide in Android Apps	SafetyNet API Check: Utilizes the SafetyNet API to assess device security. Can detect Magisk Hide by querying SafetyNet and identifying anomalies. - File System Checks[11]: Searches for Magisk-related files in the device's file system, signaling potential Magisk Hide usage. - Process and System Property Checks: Examines running processes and system properties for signs of root access manipulation, commonly associated with Magisk Hide.[12] - PackageManager Checks: Queries the PackageManager to find Magisk-related packages, even on unrooted devices. - Hook Detection: Monitors system interactions using hooking techniques and raises alerts upon unexpected changes, possibly indicating Magisk Hide presence.
SSL Pinning in Android Apps	Common SSL Pinning Libraries: Discusses libraries like OkHttp, Retrofit, and TrustKit that support SSL pinning.[13] Highlights their importance in securing network communications. - Bypassing Techniques for SSL Pinning: Explores methods attackers might use to bypass SSL pinning, including rooted devices, proxy tools, certificate manipulation, hooking techniques, and code modification.[14] - Emphasizes the need for robust SSL pinning implementation to protect against these techniques.
Code Integrity Security	- Code Hardening: Explores dynamic behavior modification to deter reverse engineering and unauthorized alterations. Provides a proactive defense against tampering attempts. - Systems Hardening: Covers tools and methodologies to minimize vulnerabilities across applications, systems, firmware, and infrastructure. - Secure Coding Practices: Includes meticulous user input validation, access controls, encryption mechanisms, and layered security to fortify code integrity. - Mobile Code Protection: Applies code hardening to safeguard mobile apps against tampering and reverse engineering, crucial for mobile security. - Software Pipeline Protection: Ensures code integrity through controlled processes, protecting the software supply chain. - Source Code Security: Involves policies, controlled access, and prevention of insecure code usage to enhance code integrity.
Decompiling Smali Code in Android Apps	Decompiling to Smali Code: Discusses the process of decompiling Android APKs into Smali code, making the code human-readable. - Analyzing Smali Code: Explores the insights gained from analyzing Smali code, understanding app architecture, logic, and security measures like SSL pinning. - Identifying Entry Points and Flow: Identifies critical methods and components that drive an app's functionality, key for potential attackers. - Rebuilding the Application: Demonstrates the potential to reconstruct apps using modified Smali code, showing how attackers can tamper with the app's behavior. - Integrating Changes: Explores how attackers can seamlessly integrate alterations into Smali code, impacting app behavior. - Compiling

ISSN (online): 2581-3048
Volume 7, Issue 11, pp 57-62, November-2023
https://doi.org/10.47001/IRJIET/2023.711009

Research focus	*Key Findings and Contributions*
	to APK: Discusses how attackers can compile modified Smali code back into a functional APK. - Implications for Security: Emphasizes the challenges in safeguarding intellectual property and user information, proposing countermeasures like code obfuscation and real-time integrity checks

This comprehensive table provides a condensed summary of the research efforts across these four Android app security domains. From detecting hidden root access to strengthening SSL pinning, ensuring code integrity, and understanding the risks of decompiling Smali code, these findings significantly contribute to the knowledge base of mobile application security. Implementing the recommendations and insights from these studies is vital for developers and organizations to build resilient Android apps in an increasingly connected and dynamic digital landscape.

V. CONCLUSION AND FUTURE WORK

The interaction of four essential elements in this all-encompassing security strategy for Android applications offers effective defense against a wide range of threats. Component 1 focuses on detecting root implementation and getting beyond Magisk Hide to protect against hazards associated with rooted devices. The focus of Component 2 is on code integrity, which improves security by spotting and preventing code tampering. Component 3 deals with SSL certificate pinning bypass, enhancing the security of data transfer. Combining knowledge from the earlier components, Component 4 orchestrates automated security implementation for financial Android apps. This approach fortifies programs against potential threats by combining root detection, code integrity, SSL pinning, and automated security, producing a strong protection architecture. This all-encompassing strategy guarantees the security of online financial transactions, protecting user information and financial assets from theft.

REFERENCES

[1] C. B. o. S. Lanka, "cbsl.gov.lk," 2020. [Online]. Available: https://www.cbsl.gov.lk/sites/default/files/cbslweb_documents/laws/cdg/psd_guideline_no_1_of_2020_e.pdf.

[2] indusface, "How to Implement Root Detection in Android Applications," [Online]. Available: https://www.indusface.com/learning/how-to-implement-root-detection-in-android-applications/.

[3] Long Nguyen-Vul, Ngoc-Tu Chau, Seongeun Kang, "Android Rooting: An Arms Race between Evasion and Detection," 29 Oct 2017. [Online]. Available: https://www.hindawi.com/journals/scn/2017/4121765/.

[4] "Detecting Magisk Hide," 4 November , 2019. [Online]. Available: https://darvincitech.wordpress.com/2019/11/04/detecting-magisk-hide/.

[5] "What is mobile application hardening?," [Online]. Available: https://cybersecurity.asee.co/mobile-application-hardening/.

[6] "The internals of Android APK build process," 7 Sep 2020.

[7] RedHunt-Labs, "Ultimate-Guide-to-SSL-Pinning-Bypass-RedHunt-Labs," [Online]. Available: https://redhuntlabs.com/wp-content/uploads/2023/07/Ultimate-Guide-to-SSL-Pinning-Bypass-RedHunt-Labs.pdf.

[8] R. Dasgupta, "Securing Mobile Applications With Cert Pinning," [Online]. Available: https://dzone.com/refcardz/securing-mobile-applications-with-cert-pinning.

[9] Francisco José Ramírez-López , Angel Jesus Varela Vaca , Jorge Ropero, "A Framework to Secure the Development and Auditing of SSL Pinning in Mobile Applications: The Case of Android Devices," November 2019.

[10] F.J. Ram´ırez-Lopez, A. J. Varela-Vaca, J. Ropero, A. Carrasco, "Guidelines Towards Secure SSL Pinning in Mobile," 2019.

[11] "How to pass SafetyNet on Android after rooting or installing a custom ROM," p. https://developer.android.com/training/safetynet/attestation, 31, MAR 2023.

[12] "SafetyNOT: On the usage of the SafetyNet Attestation API in Android," [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3458864.3466627.

[13] "Android Security: SSL Pinning," Matthew Dolan, 14, Jan 2017. [Online]. Available: https://appmattus.medium.com/android-security-ssl-pinning-1db8acb6621e.

[14] A. Bhardwaj, "SSL Pinning: Introduction & Bypass for Android," 17 April 2019. [Online]. Available: https://niiconsulting.com/checkmate/2019/04/ssl-pinning-introduction-bypass-for-android/.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 11, pp 57-62, November-2023

https://doi.org/10.47001/IRJIET/2023.711009

Citation of this Article:

R M S A Karunadasa, Y G A S Karunarathna, I A R R Illankoon, U G R M Dias, Kanishka Yapa, Amila Senarathne, "Autonomous CBSL Mobile Security Compliance Testing Tool" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 11, pp 57-62, November 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.711009

****

