The American Journal of Engineering and Technology
ISSN 2689-0984 | Open Access

TYPE Original Research
PAGE NO. 18-24
DOI 10.37547/tajet/Volume07Issue02-04

Check for updates

Data Security Methods in Mobile Applications on React Native

Ramazanov Israpil

Technical Lead, Photon Infotech, Los Angeles California US

OPEN ACCESS

SUBMITED 01 December 2024
ACCEPTED 05 January 2025
PUBLISHED 06 February 2025
VOLUME Vol.07 Issue02 2025

CITATION

Ramazanov Israpil. (2025). Data Security Methods in Mobile Applications on React Native. The American Journal of Engineering and Technology, 7(02), 18–24. https://doi.org/10.37547/tajet/Volume07Issue02-04

Abstract: The article examines methods for ensuring data protection in mobile applications developed using the React Native framework, widely employed for creating cross-platform solutions. The introduction highlights the importance of ensuring security amid the growing number of mobile device users and the increasing volume of personal data processed by applications. Developers are tasked with preventing data breaches and unauthorized access to sensitive information, which requires meticulous attention at all stages of development.

The objective of the study is to analyze data protection methods. The discussion focuses on encryption, user authentication, API security, and the use of secure storage for sensitive data. Particular attention is paid to the features of applications built with React Native and the vulnerabilities inherent to this framework. The study includes a review of existing protection methods and practical implementation of recommendations in real-world projects.

The findings indicate that a comprehensive approach is essential to ensure the security of mobile applications on React Native. Implementing biometric authentication, including fingerprint and facial recognition, as well as data encryption on both client and server sides, is mandatory. Protecting interactions to prevent attacks aimed at intercepting or modifying data is also of critical importance.

The information presented in the study will be valuable to developers, information security experts, and professionals in mobile technologies.

Keywords: Data security, mobile applications, React Native, encryption, authentication, API, biometrics, attack prevention.

Introduction: Mobile applications built on the React

Native framework are in high demand among developers due to their ability to create solutions for multiple platforms using a single codebase. The growing use of such applications is accompanied by increasing security threats, particularly in the context of the expanding number of users and the processing and storage of their personal data. This emphasizes the importance of information protection, as data breaches can lead to negative consequences for both users and developers.

These applications handle various types of sensitive information, including personal data, payment details, and passwords. Ensuring the security of this information is challenging due to the limited resources of mobile devices, vulnerabilities in client-server architectures, and the specific features of the framework. React Native is not exempt from these issues, as its unique characteristics can introduce additional risks that require tailored approaches to data protection.

The growing threats in mobile technologies demand continuous updates to security methods. The emergence of new attack types, such as data interception, information modification, and the exploitation of vulnerabilities in mobile platforms and frameworks, necessitates the improvement of existing security mechanisms. However, many available solutions fail to address the specific features of React Native, complicating the development of secure applications on this platform.

The aim of this study is to analyze data protection methods in mobile applications developed with React Native and evaluate their effectiveness.

METHODS

The article by Borawake A. V. and Shahakar M. [1] examines the use of React Native for developing an application aimed at collecting and analyzing data on the condition of dams.

The studies by Xu G. et al. and Allen J., Kelleher C. [2, 3] focus on methods for securing mobile applications that use native libraries for Android. These works describe ways to prevent leaks of sensitive information, which is particularly relevant for mobile solutions developed with React Native. The approaches presented in these studies are directed at protecting data processed through native components, making them applicable to mobile applications built on this platform.

The article by Jamarino K., Brozen M., and Blumenberg E. [6] discusses the creation of a real-time incident management system using React Native. One of the critical tasks highlighted is ensuring data security, as incidents often involve information that requires protection during transmission and storage. The study emphasizes the necessity of securing data in applications where timely responses are critical.

The work of Potocký S. and Štulrajter J. [4] is dedicated to methods for preventing the recovery of data from lost mobile devices. The proposed approaches aim to minimize the risk of leaks resulting from device loss. These methods help eliminate the compromise of personal data.

The scientific work of Munthe-Kaas H. M., Berg R. C., and Blaasvær N. [5], while not directly addressing data security issues, offers insights relevant to the development of mobile solutions. In such applications, it remains essential to consider the protection of users' personal information, even when the focus is not explicitly on this subject.

Thus, the issues of data security encompass various aspects, from preventing leaks through native libraries to ensuring the safety of real-time information in incident management systems. However, many studies lack sufficient attention to comprehensive methods for protecting data at all stages, from collection to destruction. Developing universal solutions that can be effectively applied in various scenarios for mobile applications created on the React Native platform is crucial.

The methodology of this study involves reviewing existing data protection methods and implementing recommendations in real-world projects.

RESULTS AND DISCUSSION

Modern mobile applications developed using the React Native framework face an increasing number of security threats. Although iOS and Android platforms provide security tools, their capabilities are insufficient to fully prevent data leaks, hacking, and manipulation of information [1-3]. Figure 1 below illustrates the features inherent in the architecture of React Native.

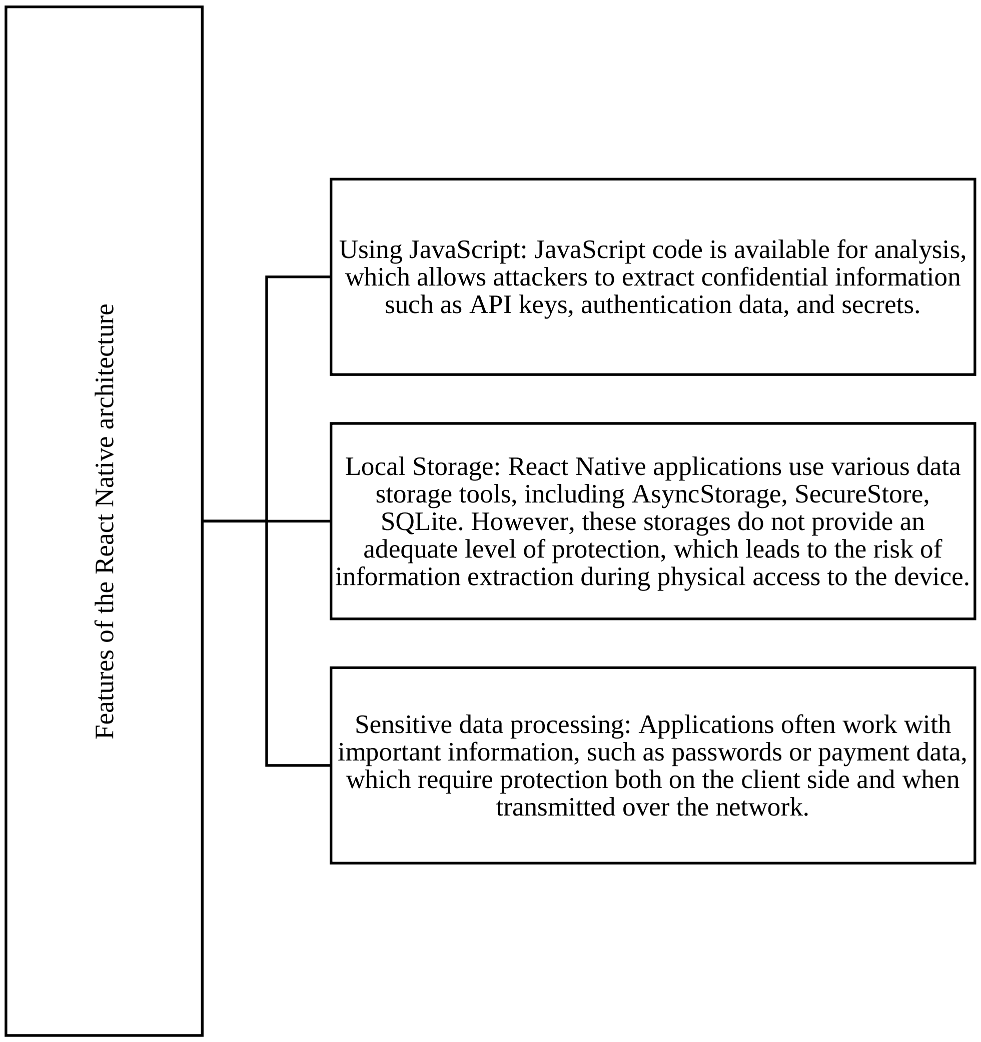
Features of the React Native architecture
Using JavaScript: JavaScript code is available for analysis, which allows attackers to extract confidential information such as API keys, authentication data, and secrets.
Local Storage: React Native applications use various data storage tools, including AsyncStorage, SecureStore, SQLite. However, these storages do not provide an adequate level of protection, which leads to the risk of information extraction during physical access to the device.
Sensitive data processing: Applications often work with important information, such as passwords or payment data, which require protection both on the client side and when transmitted over the network.

Fig. 1. Features of the React Native architecture [1-3]

Ensuring data security in mobile applications built on React Native requires a comprehensive approach, encompassing encryption, authentication, code and API protection. Adhering to security principles at every level—from data storage to data transmission—helps minimize risks of leaks, manipulation, and breaches. It is essential not only to implement these methods but also to conduct regular security audits to identify and eliminate vulnerabilities in a timely manner and to keep the application up to date.

To ensure data security in mobile applications built on React Native, several key aspects must be considered, ranging from encryption to protection against attacks and data leaks. Addressing each of these aspects creates an application with minimal risks for users [4, 6]. Table 1 below describes data protection methods.

Table 1. Data Protection Methods [4, 6]

Method Name	Description
Encryption	Encryption remains a critical method for protecting data during both storage and transmission. For React Native, encryption must cover both local storage and transmission channels. **Local Encryption**: Secure libraries such as react-native-encrypted-storage or react-native-keychain can be used to integrate with OS-secured storage like Keychain on iOS and Keystore on Android. This minimizes the risk of leaks. However, encryption alone cannot ensure complete

	security if application vulnerabilities allow access to data. **Encryption in Transmission**: Data transmitted over channels should use HTTPS with TLS. Proper certificate verification is critical to prevent "Man-in-the-Middle" attacks. SSL Pinning can be implemented to bind the application to predefined certificates, avoiding potential spoofing.
Authentication and Authorization	The process of verifying user identity must be securely organized, while controlling access to server resources. **OAuth 2.0 and JWT**: The OAuth 2.0 protocol is used for session management and authorization, separating authentication from authorization. JSON Web Tokens (JWT) are used for secure client-server data exchange without the need to store credentials on the server, reducing the risk of leaks. **Multi-Factor Authentication (MFA)**: For enhanced access security, multi-factor authentication can be implemented. This includes one-time passwords (OTP) sent via SMS or email and biometric methods like face or fingerprint recognition on iOS devices. The react-native-fingerprint-scanner library can be used to integrate biometric authentication in React Native.
Code Obfuscation and Source Code Protection	The JavaScript code used in React Native applications can be easily decompiled, posing a security risk. Attackers may extract source code to identify vulnerabilities. **Code Obfuscation**: Obfuscation alters the structure of the code, renaming variables and functions to make it harder to understand. The javascript-obfuscator tool is used for this purpose. **Use of Native Modules**: Storing sensitive information and executing critical logic in native modules enhances security, as native code is more difficult to decompile. Additionally, native modules improve application performance.
API Protection	APIs serve as the primary communication channel between the client and server, making their security a top priority. **Request Authentication**: APIs should be secured against unauthorized access using authentication mechanisms like API keys, OAuth 2.0, or JWT. Keys and tokens must not be stored in the client-side code to prevent extraction. **Rate Limiting**: To protect against DoS attacks, rate limiting is employed to restrict the frequency of requests. This helps maintain server stability under abnormal loads. **Injection Protection**: API protection must include measures to prevent injection attacks, such as SQL injections. This can be achieved through parameterized queries, ORM libraries, and server-side validation of all client-supplied data.

In turn, dependencies must be updated regularly to prevent vulnerable libraries from being exploited.

Outdated versions may contain vulnerabilities that can be exploited for attacks. Tools such as npm audit or Snyk [1, 3, 5] are used to check the security of dependencies. Next, Table 2 will describe the advantages and disadvantages of methods to ensure data security in React Native mobile applications.

Table 2. Advantages and disadvantages of data security methods in React Native mobile applications [1, 3, 5]

Security Method	Advantages	Disadvantages
1. Device Data Encryption	- Ensures confidentiality of data stored on the device.- Protects data even in case of physical access to the device.- Convenient for securing sensitive information, such as passwords and tokens.	- Impacts performance, especially when encrypting large amounts of data.- Requires additional libraries and configurations, such as react-native-encrypted-storage or react-native-keychain.
2. Biometric Authentication (Face ID, Touch ID)	- Convenient and quick authentication method, improving user experience.- Reduces the need for entering passwords or PIN codes, enhancing security.- Provides a high level of security, making biometric data hard to forge.	- Depends on device capabilities, making it unavailable on older models.- Requires external libraries, such as react-native-fingerprint-scanner.
3. Network Data Protection (HTTPS, TLS)	- Encrypts all transmitted data, preventing interception (e.g., MITM attacks).- HTTPS is a standard, ensuring secure and simple server integration.- Protects data confidentiality during client-server transmission.	- Requires proper server-side configurations to support HTTPS.- Does not eliminate the need for other layers of protection (e.g., server-side validation).
4. Token and Session Data Protection	- Secures tokens with safe storage solutions (e.g., react-native-keychain), reducing the risk of leaks.- Improves security by using secure storage (e.g., Keychain on iOS, Keystore on Android).	- Some solutions can be complex to implement and maintain across platforms.- Risk of data leakage due to improper session management or vulnerabilities in implementation.
5. Code Obfuscation	- Makes code less readable for attackers, hindering analysis and modification.- Protects intellectual property and prevents code duplication.	- Not a safeguard against all attack types, such as API exploitation.- May complicate debugging and application testing processes.
6. Secure Storage	- Ensures secure storage of data, such as tokens or passwords, in encrypted form.- Compatible with iOS and Android platforms for secure storage.	- Increases application load, requiring interaction with secure storage (e.g., Keychain or Keystore).- Does not eliminate vulnerabilities in specific device implementations or OS

		versions.
7. Role-Based Access Control (RBAC)	- Controls access to application functionality based on user roles.- Effective in multi-user applications requiring different access levels.	- Requires setup on both client and server sides.- Increases development and testing complexity.
8. Regular Updates and Security Patches	- Regular application and library updates protect against emerging threats.- Reduces the risk of vulnerabilities in older versions.	- Requires continuous monitoring and implementation of new patches, making the process labor-intensive.- Delays in updates can expose the application to vulnerabilities.
9. Two-Factor Authentication (2FA)	- Provides an additional layer of security for user accounts.- Makes unauthorized access more difficult, even in cases of password leaks.	- Requires extra steps for users, potentially reducing convenience.- Integration with external services (e.g., SMS or authentication apps) is necessary.
10. Code Injection Protection (e.g., SQL Injection)	- Protects against attacks involving malicious code in server queries.- Reduces the risk of data corruption and system compromise.	- Requires validation and protection for all server interactions, adding development complexity.- Not always able to prevent all injection types, especially with inadequate input data control.

Thus, ensuring the security of mobile applications developed with React Native requires a comprehensive approach and adherence to various best practices. Data protection during transmission, storage, and processing, as well as protection against attacks, including MITM, injections, and data leaks, must all be considered to create a reliable and secure application.

CONCLUSION

This study analyzed data protection methods in mobile applications developed using the React Native framework. The relevance of the topic is determined by the increasing security threats associated with this platform. The analysis revealed that to ensure reliable protection of user information, it is necessary to consider not only standard approaches, such as encryption, authentication, and API security, but also the specific characteristics of mobile operating systems on which the applications operate.

Attention was given to vulnerabilities specific to React Native and measures aimed at minimizing them. It is crucial to establish secure interactions between client and server components, apply modern data encryption methods for both storage and transmission, and implement biometric technologies to strengthen authentication.

REFERENCES

Borawake A.V., Shahakar M. Embankment Protection - A cross-platform React Native application for embankment protection using crowdsourcing data //The 2021 International Conference on Computing, Communication and Green Engineering (CCGE). – IEEE, 2021. – pp. 1-7.

Xu G. et al. SoProtector: Privacy protection of proprietary SO files in developing mobile applications of the Internet of Things //IEEE Internet of Things Journal. – 2019. – vol. 7. – No. 4. – pp. 2539-2552.

Allen J., Kelleher S. The viability of React examples for effective API Learning (REVEAL): a tool to help programmers use incompatible code examples in React. js //Journal of Computer Languages. – 2023. – Vol. 75. – p. 101201.

Potocký S., Štulrajter J. Advanced Anti-Forensic Protection of Mobile Applications //2023 Communication and Information Technologies (KIT). – IEEE, 2023. – pp. 1-8.

Munte-Kaas H. M., Berg R. S., Blaasver N. The effectiveness of measures to reduce homelessness: a systematic review and meta-analysis //Campbell Systematic Reviews. – 2018. – Vol. 14. – No. 1. – pp. 1-281.

Jamarino K., Brozen M., Blumenberg E. Planning in support of and against the automotive Homelessness: Spatial trends and determinants of motor housing in Los Angeles //Journal of the American Planning Association. – 2023. – Vol. 89. – No. 1. – pp. 80-92.