# Case Study Report: Insecure Mobile App Development

## Introduction

With the convenience of mobile apps comes significant security challenges, especially when applications handle sensitive information. One of the most critical vulnerabilities in mobile app development arises from the improper storage of sensitive data. This report explores the risks associated with such vulnerabilities, the causes behind them, and the strategies to secure mobile app development. **Insecure banking applications** highlight the consequences of neglecting proper data security practices.

## Problem Analysis

In this scenario the core issue is surrounding the storage of sensitive data in plain text on mobile devices. This form of storage provides no form of protection against any unauthorized access, meaning attackers can extract the data if the device is compromised, stolen or infected with malware.

It is important to have encryption mechanisms in place to protect the stored data. Encryption is fundamental for security that transforms data into an unreadable format which can only be read using a decryption key. The app did not have authentication methods, such as a multi-factor authentication which would have added an extra layer of security even if the data was compromised.

There are several risks which are associated with a vulnerability:

1. **Financial Fraud:** An attacker could use stolen credentials to perform unauthorised transactions.
2. **Identity Theft:** The user’s sensitive information to perform fraudulent activities.
3. **Reputational Damage:** Trust between the user and the company will be damaged if the security fails, leading to a decline in the adoption of the app.
4. **Regulatory Penalties:** If sensitive data is stored insecurely this will violate the data protection regulations, such as GDPR or PCI-DSS. This will result in legal and financial repercussions for the organisation.

### Root Causes

Several key factors which cause these vulnerabilities:

**Plain Text Data Storage –** When the organisation stores its user credentials and transaction data in plain text. This shows a lack of adherence to the practices that should be considered when securing data.

**No Encryption –** This is when proper encryption mechanisms have left sensitive information exposed to unauthorised access.

**Lack of Secure Authentication –** The use of basic authentication methods, such as multi-factor identification, were used without the proper additional layers of security. Such methods may include MFA or even a biometric scanner.

**Insufficient Security Awareness –** During the development process functionality and speed may have been prioritized over security. This oversight is where the vulnerabilities lay and through which attackers gain access to sensitive data.

### Consequences

There are several consequences associated with insecure data storage in mobile apps.

**User Risks –** It is the users who face the most immediate risk of threats. These risks could be anything from financial loss to identity theft. Attackers may be able to use stolen credentials to commit fraudulent activities under the user’s identity.

**Organizational Risks –** If an organization is found to have been lacking in its data storage, then this would lead to reputational damage as its users will have lost trust in the app's security. With this also comes legal liabilities, fines and an overall decline in the app’s user base.

## Proposed Solutions

**Secure Data Storage**

The most critical step is to eliminate plain text storage of sensitive data. Sensitive information should all be encrypted using industry-standard algorithms such as AES-256. By doing this the data will remain unreadable without the appropriate decryption key and protected against attackers.

There are also secure storage mechanisms which are provided by mobile operating systems which should be leveraged by developers.

* iOS: Utilize Keychain Services.
* Android: Use EncryptedSharedPreferences or Keystore AP1.

WhatsApp, for example, uses end-to-end encryption which ensures all messages are encrypted before they leave the device.

**Enhanced Authentication**

Another additional layer of security can be added by implementing Multi-Factor Authentication. If users are required to verify their identity through multiple methods this will reduce the risks of unauthorized access to user data. Such MFA methods include apps like Microsoft Authenticator, or a one-time code sent to a mobile device. Biometric authentication adds an even more secure option which is thought to be a more user-friendly option. Examples for such include fingerprint identification or facial recognition. Google employs such biometric authentication to protect transactions.

**Secure Transmission**

Man-in-the-middle attacks can be prevented using HTTPS with TLS when data is being transmitted between the app and external server. Further adding to that would be implementing certificate pinning to ensure the app only communicates with trusted servers.

**Regular Security Audits and Updates**

By doing routine security audits and penetration testing early in the development process, organizations can identify vulnerabilities. Apps should also include mechanisms that allow for regular updates to patch any vulnerabilities that may emerge post-deployment. With the user's consent, these automatic updates can ensure the adoption of security fixes.

**User Education**

There should be clear and accessible documentation, or an interactive tutorial provided to the user on practices surrounding technical measures. This would educate the user on matters such as creating a strong password and enabling security features within the app.

## Implementation Example

**2017 Equifax Data Breach**

One of the world’s largest data breaches occurred in 2017 which affected Equifax, one of the three major credit reporting agencies. The breach exposed the personal data of approximately 147 million users. It resulted from Equifax’s failure to apply a security patch for a vulnerability in Apache Struts, a software framework used to create enterprise-ready Java applications.

The vulnerability had been identified and patched in March 2017, but Equifax failed to apply the patch for several months, leaving their system exposed. Attackers exploited this vulnerability to access sensitive user data, including names, Social Security Numbers, birthdates, addresses, and driver’s license numbers.

The breach is particularly relevant to the banking sector because financial institutions often rely on credit reports and data from agencies like Equifax to verify identities and assess creditworthiness. With this information exposed, attackers could gain access to user accounts and commit identity theft.

Those customers who used their mobile banking to check credit scores or even to apply for loans may also have had their data exposed. Using this information, hackers could then use the stolen data to make fraudulent transactions.

There are several consequences for both the user and the bank associated with the vulnerability such as:

**Identity Theft**

Once the attackers have gained access to the user's personal data, they can then impersonate them to make fraudulent transactions in the customer's name. With the Equifax data breach, attackers were able to commit crimes such as opening new lines of credit and making unauthorized withdrawals.

**Financial Loss**

Those customers whose data was stolen faced the risk of financial loss if the attackers used the data to commit fraud. Equifax faced several lawsuits and ended up paying a settlement to the Federal Trade Commission, the Consumer Financial Protection Bureau, and 50 U.S. states and territories which totalled $700 million. The company's stock prices also plummeted after the breach was announced.

**Reputational Damage**

The stock dropping showed the loss of stockholder faith and trust in the company. They also faced scrutiny from several regulatory bodies which led to investigations being conducted into Equifax’s security practices and the measures taken to protect customer data.

## Banking Apps and Secure Programming Lessons

**Timely Patch Management**

It is important for financial institutions that use the likes of mobile apps to apply security patches promptly. By performing regular software updates these companies can avoid such vulnerabilities being exploited by attackers.

**Encryption**

Customer data must be encrypted both in transit and at rest by such companies. This will ensure that even if attackers did manage to gain access to the data, they would not have the decryption keys.

**Multi-Factor Authentication**

Banks who use mobile banking apps require additional layers of authentication layers so that only the rightful owner of the account can access their personal data. Such authentication layers may include SMS verification, fingerprint scanning or face recognition.

**Secure APIs**

By using the likes of Equifax, mobile banking apps should ensure that these external API calls which are made between the app and the third party are secured. They would need to be secured using encryption tools like QAuth for the authentication process.

**User Education**

Using their mobile apps, banks should make their users aware of the different phishing techniques used by hackers. By providing these guidelines to its customers, the company educates its users on how to further secure their accounts and prevent infiltration or data breaches.

## Discussion and Broader Implications

The security vulnerabilities identified in the case study extend beyond mobile banking apps, affecting industries like e-commerce and healthcare that rely on platforms like Equifax to manage sensitive data. The urgency to release applications often leads to neglected security features, leaving systems vulnerable to data breaches.

Given the Equifax breach, companies using mobile apps should adopt a security-first approach during software development. Security must be integrated at every stage of the development lifecycle. Frameworks like DevSecOps enable continuous security integration, while secure coding standards, regular penetration testing, and comprehensive risk assessments help identify and address vulnerabilities early.

Frameworks like GDPR and PCIDSS should enforce best practices for cybersecurity more consistently, as inconsistent enforcement across jurisdictions weakens their effectiveness.

## Conclusion

This case study underscores the critical need for robust security practices in mobile app development, with the Equifax data breach serving as a powerful example of the severe consequences of poor data protection. To mitigate the risks, organizations must adopt a security-first approach, ensuring data encryption, enhanced authentication methods, and regular security audits. Leveraging secure storage mechanisms provided by mobile application systems and educating users on safe practices further strengthens security. Ultimately, a collaborative effort between developers, businesses, regulators, and users is essential to safeguarding sensitive information and maintaining trust.

# References

*About end-to-end encryption | WhatsApp Help Center* (no date). Available at: https://faq.whatsapp.com/820124435853543 (Accessed: 30 January 2025).

*Authentication tools for secure sign-in – Google Safety Centre* (no date). Available at: https://safety.google/authentication/ (Accessed: 30 January 2025).

*Chinese Hackers Charged in Equifax Breach — FBI* (2020). Available at: https://www.fbi.gov/news/stories/chinese-hackers-charged-in-equifax-breach-021020 (Accessed: 26 February 2025).

*Financial watchdog fines Equifax Ltd £11 million for role in one of the largest cyber security breaches in history* (2023) *FCA*. Available at: https://www.fca.org.uk/news/press-releases/equifax-ltd-fine-cyber-security-breach (Accessed: 26 February 2025).

Kananda, V. (2022) *What is AES 256 Encryption & How Does it Work?* Available at: https://www.progress.com/blogs/use-aes-256-encryption-secure-data (Accessed: 28 January 2025).

*Keychain services* (no date) *Apple Developer Documentation*. Available at: https://developer.apple.com/documentation/security/keychain-services (Accessed: 28 January 2025).

*KeyStore (Java Platform SE 7 )* (no date). Available at: https://docs.oracle.com/javase/7/docs/api/java/security/KeyStore.html (Accessed: 28 January 2025).

*What is: Multifactor Authentication - Microsoft Support* (no date). Available at: https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661 (Accessed: 28 January 2025).

# Document Stats

The case study itself (pages 1-5) contains approximately 1,604 words.

Font type: Arial

Font Size: 11

Page count: 7

References: Harvard Style (Zotero generated)