Securing Mobile Applications

Best practices in Mobile Application Development

Edited by **MOHAN DAS VISWAM**

Android, iOS and hybrid apps are vulnerable to a range of threats, and businesses need to be protected from the risk associated with running mobile apps in an unprotected environment. This becomes more important when it is about the Apps for government. Platform-specific security best practices must be followed for robust and secure mobile application development.

T Mohana Dhas
Dy. Director General & SIO
mohandhas.t@nic.in
sio-ker@nic.in

Manoj P A
Sr. Technical Director
manoj.pa@nic.in

Andrews Varghese
Technical Director
andrews.varghese@nic.in

Syamkrishna B G
Scientist B
syam.krishna@nic.in

Internet and mobile usage in India is all set to cross the 900-and mobile usage in India is all set to cross the 900-million mark by 2023, with nearly two-thirds of the population estimated to have Internet access and a mobile device. Mobile apps are becoming the main medium of digital interaction. Modern-day users are on the move and they are utilizing mobile applications for most of the electronic transactions.

Mobile Application Threats

A smartphone user is exposed to various threats when they use their phone. Attackers exploit weaknesses inherent in smartphones or flaws in the application logic. Mobile apps are often the cause of unintentional mass data leakage. Client-side vulnerabilities can be exploited without physical access to the phone. Improper platform usage has been the leading mobile security vulnerability, which refers to the misuse of any platform-specific feature or failure to incorporate platform security controls. Mobile applications developed with default configurations are vulnerable to certain known attacks. It is important to know such vulnerabilities while developing mobile apps. Another most common vulnerability is insecure data storage. If an attacker can physically access the phone, the attacker can copy application data to a computer.

A mobile application resides completely in the user device. There are free tools available to decompile and regenerate source codes of a mobile application. An attacker can learn the business logic from the decompiled source code and can attack the IT infrastructure using the credentials taken from the application installed in the user device. Securely storing the secrets in a device is also very important for securing the IT infrastructure from various attacks. Platform-specific security best practices must be followed for robust and secure mobile application development.

Developer Challenges

Along with security, the privacy of user information is the primary concern for an app developer. A hacker/ intruder should be prevented from getting access to the critical data of the mobile application. The app must protect the following from a hacker/ intruder:

- Private, sensitive, and personal information
- Unauthorized access to the system
- Execution in a rooted/ jailbroken environment

A malicious user or hacker can use a rooted/ jailbroken device to install the application to study the logic and API information. The attacker can even create malicious/ fake apps targeting the APIs.

Mobile applications can be login based or without login, based on the usage and security of services provided by the enterprise. Login based applications can have the following authentication mechanisms in place, with one or more factor for authentication (specifically, what a user knows, has, and is), to determine the user's identity.

- Login with a username and password
- LDAP authentication
- Login using a unique Id provided by the enterprise (e.g. Aadhaar) along with two-factor authentication/ N- Factor (OTP, Biometric etc.)
- Social Media login (Login with known OAuth APIs of Google, Facebook, Twitter etc.)

To address the security concerns, following are the major challenges:

- Web API security
- Securely storing the secrets

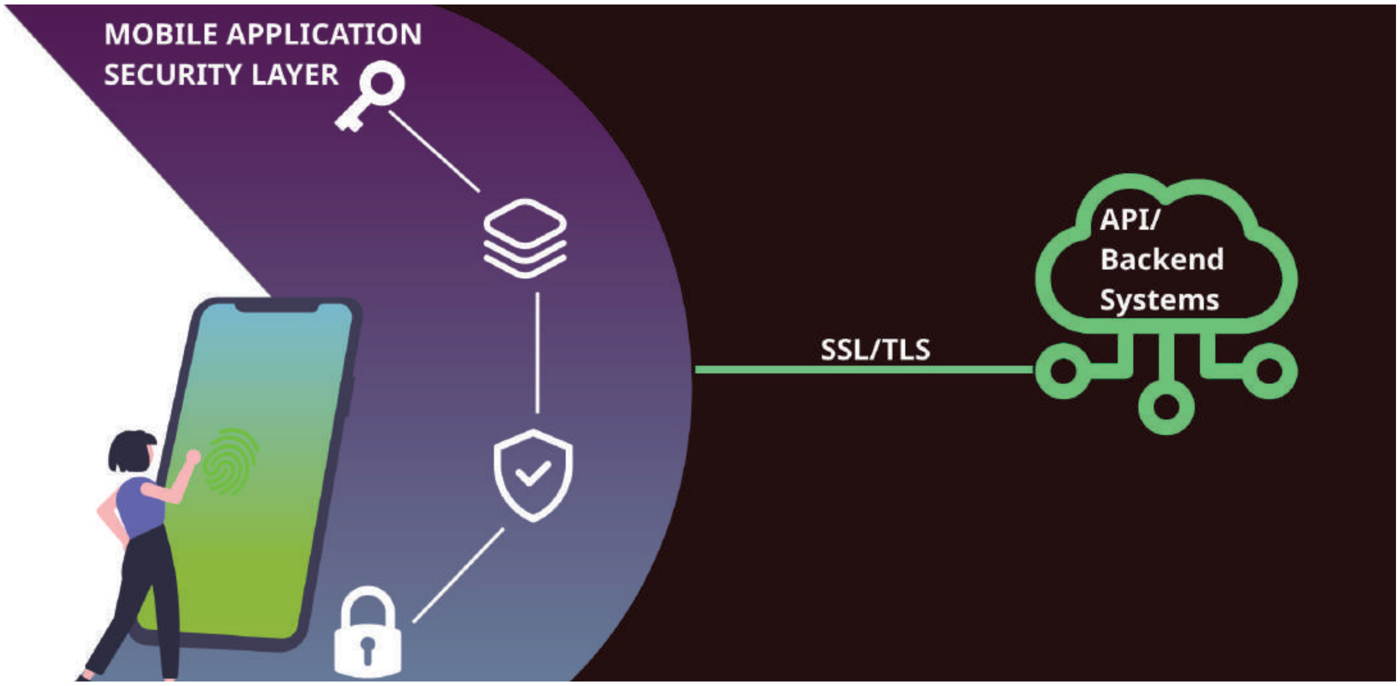
- Restrict running in rooted/ jailbroken device

Web API security

Mobile apps use APIs to interact with backend systems. API keys and tokens play an important role in application security, efficiency, and usage tracking. Best practices must be followed while selecting strong encryption standards. Transport Level Security (TLS) is a standard approach for securing an HTTP channel. To ensure message integrity, a message authentication code (MAC) for each request using the shared secret with an algorithm such as HMAC SHA-256 is recommended.

Storing Secrets

There should be a minimum-security mechanism in place for APIs used by remote non-sandboxed clients like mobile applications. Security of the APIs is partially dependent on the secure integration by the app. Different schemes can be in place to design a secured architecture. The factors affecting the security scheme depend on the data and business environments.



When a mobile app runs on a user device, it is necessary to store some user preferences and related configuration information in the user device itself to provide a seamless experience to the application user. It is very important to store such information securely on the user device. For example, the API keys for a web service, sensitive private and confidential information etc. It is always better to utilize the most secured storage option provided by the device operating system.

API keys and other sensitive information should be encrypted and stored in the device. The data can be encrypted using a key known to the client or encoded using an encoding scheme. The level of encryption can be decided based on the business and service provided by the enterprise through the application. Encryption logic can be built, on-demand, based on the following:

- **User provided PIN**
- **Keys shared through a secondary channel (OTP shared as SMS/ EMAIL)**
- **Keys securely stored in KeyStore**
- **Keys stored within the application code (hard-coded)**

It is not recommended to hardcode important application URLs in the application itself. Such URLs should be supplied to the application at run time only. Hardcoded values must be stored as byte arrays and can be converted to strings or of required data types at run time. It is recommended to store such hardcoded values in native layers.

The Encrypted Shared Preferences class is available in the Android Jetpack library. This library uses device-specific features for securely storing user configurations. Android Keystore mechanism also can be used to create keys that can be used for encryption purposes. A named key can be created in keystore, which by default is accessible for the app which created the named key, which can be used for device-level encryption and decryption. Best practices must be followed to make the process faster as cryptographic operations are normally time-consuming. In the case of iOS applications, a keystore must be used for securely storing application-specific secrets.

App in a rooted/ jail broken device

Jailbreaking is the process of removing software restrictions put into place by Apple on devices that run the iOS operating system. Similarly rooting in Android is the process of removing software restrictions put into place by Google and gaining the ability to access the entire operating system. A legitimate app running on a jailbroken or rooted mobile device is more vulnerable as it can expose sensitive user data. Platform-specific methods are there to detect a jailbroken or rooted phone. There is no one-size-fits-all solution for detecting jailbroken or rooted devices. There are attestation service providers who remotely evaluate the devices, whether the request is coming from the genuine app running on a genuine Android device. Attestation services may be used in the application if the business demands it.

Hiding Business Logic

Obfuscation mechanisms make it difficult to understand business logic. In software, the obfuscation of code is the process of modifying an executable so that it is no longer useful to unauthorized parties such as hackers but remains fully functional. Mobile application code, wherever possible, must be obfuscated before deployment. There are a few tools related to Android Studio such as R8, ProGuard and DexGuard. R8 is a free tool that is included in Android Studio.

Another method is to use NDK in android and write business logic/ codes using C/C++. Decompiling and getting the source codes from NDK compiled files are very difficult.

User awareness

Security depends on users. It is also important to educate the end-users regarding vulnerabilities that can be caused by installing apps from untrusted sources. Even legitimate apps from reliable marketplaces can include high-risk security issues. These apps can steal user information and configuration from other applications installed in the device. The security implementation in any system needs revisit and improvements regularly with increasing threats in the cyber world. Device owners must take responsibility for protecting the data they store in mobile applications. But user precautions will still fall short if developers leave vulnerabilities in their applications.

Mobile app security is the measure and means of defending mobile applications from digital fraud in the form of malware, hacking, and other criminal manipulation. The OWASP Mobile Security top 10 (https://www.owasp.org) may be referred for more awareness about the current mobile security issues.

References

Centre for Competence for Mobile App Development Kerala has published a document titled 'Secure API Integration for Mobile Apps'. The document explains use cases and best practices along with methods for securely storing secrets in the device. This document is available in the digitalNIC Platform of NIC.

For further information, please contact:
STATE INFORMATICS OFFICER
NIC Kerala State Centre
CDAC Building, Vellayamabalam
Thiruvananthapuram - 695033
KERALA
Email: sio-ker@nic.in, Phone: 0497-2700761