Do *you* know Mobile Security?

Securing the backend

- Enforce strong authorization and authentication policies on the backend.
- Treat all input from the client as untrusted; validate and sanitize the data.
- Use parameterized queries to communicate with backend data stores.

Securing data storage

iOS

- Use an iOS encryption library to protect cached info.
- For example, the Keychain Services API that provides secure storage for passwords, keys, certificates, etc.
- Use SQLcipher to encrypt SQLite databases.
- Don't use NSUserDefaults or CoreData to store sensitive information - they save data to an unencrypted plist file on the device.

Android

- Encrypt the app data storage area on the device.
- Use setStorageEncryption in the DevicePolicyManager class in the android.app.admin API.
- Store app-only keys in the Android Keystore.
- Store system-wide keys in the Android Keychain.
- Encrypt the app's database on the device.

Protect the transport layer

- Encrypt all communication between app & backend with TLS at 128-bit encryption or higher.
- Use certificate pinning for TLS connections to authenticate connections to the backend.
- Only use trusted certificates and don't bypass verification.
- For iOS, allow iOS to manage the TLS verification.

Unintended data leakage

- Disable the keyboard cache, third-party keyboards & clipboards.
- Watch out for data leaks in keyboard caches, background snapshots, clipboards, logs, user dictionaries, and screen sharing apps.
- Do not log sensitive actions.
- For Android, Use FLAG_SECURE in LayoutParams for any Activity that displays sensitive info.

Authentication and authorization

- Authenticate and verify all requests on the backend server, never in the application.
- Avoid storing credentials; when not possible, store data using keystores and encryption.
- Consider using OAUTH2 or appsecret tokens. If not, store tokens on the backend.

Client-side injection

iOS

- Display external links in Safari, not in the app using UIWebView.
- Link to external sites through HTTPS.
- Use loadHTMLString with baseURL as about:blank instead of loadRequest, which allows file access.
- Remember do not base security decisions on untrusted inputs and escape characters in URLs, HTML, HTML attributes, JavaScript, etc.

Android

- Parameterize queries that go through WebView.
- Disable JavaScript and plugin support in WebView.
- Disable WebView access to files and disable WebView for apps that don't need to launch web pages.
- Stop external callers from launching app component: set android:exported=false in the manifest file.
- Validate all intent actions.
- Explicitly define all component permissions.
- Run Broadcast Receivers only when needed: registerReceiver to start, unregisterReceiver to end.

Use cryptography

- Use crypto libraries deployed with the mobile framework.
- Do not use your own algorithms.
- Be cautious with key management and do not hardcode crypto primitives.
- Prefer Perfect Forward Secrecy for TLS connections.
- Wipe crypto primitives from memory after use.

Security decisions

- Do not pass sensitive info via IPC and do not accept sensitive data or changes through URL schemes.

iOS

- Parameterize all input.
- Pass sensitive info through iOS Keychain Access Groups.

Android

- Parameterize all input.
- Configure secure access to and permissions for Intents, Intent filters and other components.

Use session management

- Always use session timeouts.
- Require re-authentication after session timeout. Don't re-use tokens for new sessions, always issue new ones. Remember to expire and invalidate old tokens.

Use binary protections

iOS

- Always obfuscate code.
- Detect debuggers by checking the CPU kernel's process list for a ptrace flag using AmIBeingDebugged.

Android

- Always obfuscate code.
- Use root verification.
- Use code signing to detect tampering.
- Shut the app down if tampering is detected.

Want to learn more about Mobile Security?

Get a deeper understanding of core security concepts in both iOS and Android operating systems. Dive into more! Enroll in our **Defending iOS** and **Defending Android** course today. Contact us at info@securitycompass.com.

www.securitycompass.com/training

Security Compass