Category	ToCheck list	Tools
Information Gathering	General Information. List of general application data	
	Components list. List of application components.	
	Component permissions. List of application component permissions.	
	Components exported.	
	Component Launching Intents. List of component launching intents.	
Configuration And Deploy management	Debugging mode left enabled	
	Vulnerable 3rd party libraries still in use	
	List of backup files stored in application	
	List of metadata applicable to file within the application.	
	List of permissions of the files created by the application.	
	List of Content Provider component permissions.	
	List of Activities component permissions	
	List of Services component permissions.	
	List of Broadcast Receivers component permissions.	
	List of permissions of the databases created by the application.	
	List of permissions of the shared preferences created by the application.	
Authentication	Default Users and Passwords.List of default users and passwords stored by the application.	
	Weak Password Policy. Weaknesses related with the password robust policies.	
	If the app provides users with access to a remote service, an acceptable form of authentication such as username/password authentication is performed at the remote endpoint.	
	A password policy exists and is enforced at the remote endpoint.	
	If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user's credentials.	
	If stateless token-based authentication is used, the server provides a token signed using a secure algorithm.	
	The remote endpoint terminates the existing stateful session or invalidates the stateless session token when the user logs out.	
	Biometric authentication, if any, is not event-bound (i.e. using an API that simply returns "true" or "false"). Instead, it is based on unlocking the Keystore.	
	The remote endpoint implements an exponential back-off, or temporarily locks the user account, when incorrect authentication credentials are submitted an excessive number of times.	
	Remember Credentials Functionality. Weaknesses related with credential storage through the remember functions.	
Cryptography	Hard-coded credentials.Passwords stored within the source code.	
	Insecure Data Storage.Weaknesses related with confidential information storage.	
	Insecure use of Transport Protocol. Weaknesses related with unsecure information flow.	
	The app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices.	
	All random values are generated using a sufficiently secure random number generator.	
	The app doesn't re-use the same cryptographic key for multiple purposes.	
	Certificate Pinning. Weaknesses related with trusted chains of digital certifications.	
Information Leak	Information Leak to log files.	
	Information Leak to SDCard.	

Information Leak	Information Leak to Network	
	Information Leak to Android’s IPC	
Input Data Validation	Cross Site Scripting	
	Buffer Overflow	
	SQL Injection	
	Path Injection in File Access	
	Null Parameter Checking	
	Log Injection	
	Injection Process Control via Intent Data	
Intent Spoofing	Intent Spoofing on Broadcast components.	
	Malicious ActivityLaunch	
	Malicious ServiceLaunch	
	Weaknesses related with the insecure use of Pending Intents.	
Unauthorized Intent Receipt	Intent interception on Broadcast components.	
	Intent interception on Activity components	
	Intent interception on Service components.	
	Pending Intent interception.	
Business Logic	Business Logic	
Data Storage	The Keystore is used to store sensitive data, such as user credentials or cryptographic keys.	
	No sensitive data is written to application logs.	
	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	
	The keyboard cache is disabled on text inputs that process sensitive data.	
	The clipboard is deactivated on text fields that may contain sensitive data.	
	No sensitive data is exposed via IPC mechanisms.	
	No sensitive data, such as passwords or pins, is exposed through the user interface.	
Platform Interaction	The app only requests the minimum set of permissions necessary.	
	All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.	
	The app does not export sensitive functionality via custom URL schemes without proper protection.	
	The app does not export sensitive functionality through IPC facilities without proper protection.	
	The app does not export sensitive functionality through IPC facilities without proper protection.	
	WebViews are configured to allow only the minimum set of protocol handlers required (ideally, only https is supported). Potentially dangerous handlers, such as file, tel and app-id, are disabled.	
	Object serialization, if any, is implemented using safe serialization APIs.	
Network	Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.	
	The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.	

	The app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a trusted CA are accepted.	
Code Quality	The app is signed and provisioned with valid certificate.	
	The app has been built in release mode, with settings appropriate for a release build (e.g. non-debuggable).	
	Debugging symbols have been removed from native binaries.	
	Debugging code has been removed, and the app does not log verbose errors or debugging messages.	
	The app catches and handles possible exceptions.	
	Error handling logic in security controls denies access by default.	
	In unmanaged code, memory is allocated, freed and used securely.	
	Free security features offered by the toolchain, such as byte-code minification, stack protection, PIE support and automatic reference counting, are activated.	
Defense In Depth	No sensitive data is included in backups generated by the mobile operating system.	
	The app removes sensitive data from views when backgrounded.	
	The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.	
	The app enforces a minimum device-access-security policy, such as requiring the user to set a device passcode.	
	The app educates the user about the types of personally identifiable information.	
	A second factor of authentication exists at the remote endpoint and the 2FA requirement is consistently enforced.	
	Step-up authentication is required to enable actions that deal with sensitive data or transactions.	
	Sessions and access tokens are invalidated at the remote endpoint after a predefined period of inactivity.	
	The app informs the user of all login activities with his or her account. Users are able view a list of devices used to access the account, and to block specific devices.	
	The app either uses its own certificate store, or pins the endpoint certificate or public key, and subsequently does not establish connections with endpoints that offer a different certificate or key, even if signed by a trusted CA.	
	The app doesn't rely on a single insecure communication channel (email or SMS) for critical operations, such as enrollments and account recovery.	
	The app detects whether it is being executed on a rooted device. Depending on the business requirement, users are warned, or the app is terminated if the device is rooted.	
Static Analysis	Manifest file giving un-necessary permission	
	Test for sensitive info in Jar files	
	Code Analysis with mobilizer	
	Test for sensitive info in SMALI files	
	Test for exported activity component	
	Test for exported Broadcast receivers	
	Test for sensitive data into shared preferences	
	Test for exported provider component	
	Insecure Shared Storage	
	Test for un-encrypted local database files	
	Test for sensitive info in clear text in internal storage	
	URL Caching (HTTP Request and Response) on cache.db	
	Copy/Paste Buffer Caching	
	Keyboard Press Caching	
	Test for sensitive info in clear text in external storage	
	Test for sensitive info being saved inside user dictionary cache	

Dynamic Analysis	Test for sensitive info in Heap Memory dump hex	
	Test for android WebView implementations and JavaScript support	
	Test for background process running with the application	
	Test for cookies attributes	
	Test for vulnerable file uploads	
	Unauthorized Code Modification	
	Test for data encryption implementation	
Root Detection checks	Test if the app checks fro root detection on device or not and then check if it is possible to bypass it	
UI Security	Test all the application views (activities) and check if it might be vulnerable to Tap jacking.	
	Test for any client side based authorization which might be bypassed.	
Execution of Untrusted Code	Exposing External Java Interfaces in WebViews DOM	
	Code Signing	
	Loading Dynamic DEX onto Dalvik	
	Abusing Dynamic Code Execution Decisions	
	Object Lifetime Vulnerabilities (Use-after-free, double free's)	
	Format Strings Vulnerabilities	
	NDK Exposed Code Secrets	
	Integer Overflows	
	Integer Underflows	
Transport Layer Security	Insecure Transport Layer Protocols	
	TLS Authenticity Flaws	
	TLS Weak Encryption	
	Bypassing TLS Certificate Pinning	
	TLS Known Issues – CRIME, BREACH, BEAST, Lucky13, RC4, etc…	
	SSL/TLS renegotiation attack	
	Disable certificate validation	