ADHD Task Manager

Security Review

Overview

This document will review all aspects of the ADHD Task Manager applications current security practices Vs the recommended by Google best practices for application security for Android applications. The document the changes that need to be implemented in order for our application to follow the Android security best practices.

Delaying Sensitive Data Being Loaded In

Google recommends that applications delay sensitive data being loaded in until after the user has signed in. Currently we have are application set up so that the databases that are stored locally on the users device are encrypted, however, the databases are decrypted once the application is started.

To fix this security issue, we will need to have the application await the user successfully signing in before decrypting the local databases. There are several ways we coud do this, we could use a simple conditional check to see if the user has successfully signed in, or we could use a launched effect to await the change in the current user state. My preference is to go with the latter option, using a launched effect to await the users successful sign in before running the logic that will decrypt the local databases.

WebView Security

Google recommends that applications do not allow users to use a WebVeiw in their application to navigate to any sites that are outside of their control. They also recommend that applications do **NOT** enable JavaScript interface support unless applications can completely control and trust the content in the applications WebView.

Currently our application WebView has JavaScript interface support enabled, but does not have any restrictions on what content can be accessed in the WebView. To fix this security issue we will need to block any content that we do not control or trust. We can do this this by using both an allowlist, as recommended by Google, and by blocking loading content from the network, which we can do because the HTML pages for the Help Page WebView are packaged with the application and accessed locally.

Use The Right Permissions

Google recommends using the minimum number of permissions possible, and wherever possible, to use an intent to defer the request to an app that already has the required permissions. The example they provide is using an intent to defer the request to the users contacts application instead of requesting the READ_CONTACTS and WRITE_CONTACTS permissions.

In this regard our applications is already following Googles recommendations, as we do not request permissions that are not absolutely necessary for the functionality of our application. Additionally Android allows applications to read and write files to the users device without requesting permissions from the user, this is because the reading and writing of files is handled by the Android operating system.

Store Local Data Securely

Google recommends that data that is stored locally on the users device is encrypted with AES256 encryption to keep the data secure. Our application is following this recommendation as both the Todo database and Reward database are encrypted with AES256 encryption.

Checking Validity Of Data

Google recommends that when an application uses an external storage source, as our application does, that the data is checked for corrupted or modified data. Their recommendation for doing this is to use a hash verifier so that your application can store hash codes securely locally and compare the hash codes to the data that is received by the external database. Google also recommends that the hashing function should **NOT** be done on the main thread and should be done on a separate thread because generating hash codes can take some time.

Currently our application does not generate, store and compare hash codes for data that is sent and received to and from the external database. We will need to develop a hashing function before we deploy our application to the Google Play Store.

Keeping Services And Dependencies Up To Date

Google recommends keeping all services, dependencies and external libraries up to date throughout the life of the application to help keep the application secure.

Our application is currently using the most up to date dependencies, services and external libraries as of the time of writing this report, and we will continue to up date them as required into the future.

Storing API Keys Securely

Google recommends that applications use KeyStore to store required API keys securely. Currently our application is using BuildConfig to store these values, however, we will need to move to using KeyStore before launching our application to the Google Play Store.

Identified Security Risk And Thier Current Status

Security Risk	Current Status
Delaying Sensitive Data Loading	Moderate Risk
WebView Security	High Risk
Use the right permissions	No Risk
Store local data securely	No Risk
Checking validity of data	High Risk
Keeping services and dependencies up to date	No Risk
Storing API keys securely	High Risk