AarogyaSetu - Technical FAQs[1]

While utmost attention has been paid to every aspect of the AarogyaSetu source code, so as to avoid any security lapses, yet some source code analyzers or scanners, may report some possible security issues in the App. The possible impact of such issues has been studied by the AarogyaSetu team and the following clarifications are offered in this context for the better understanding of the overall security community.

The following issues and clarifications should be read in the context of a normal android user, who may use the App on a non-rooted phone with debugging disabled. The security community can review these clarifications in the context of AarogyaSetu Application and if they have any contrary views, or if they were able to find any possible way to exploit these issues (on a non-rooted phone without bypassing SSL pinning and without using Android Debugging), then the same can be brought to the notice of the AarogyaSetu Team by sending a mail to : as-bugbounty@nic.in

What is considered as Hacking and What is not

What is considered as hacking the App ?

Finding an ability to access an user's personal information (other than openly broadcast Bluetooth DiD information) in proximity of the user's phone or remotely - without the user havingcompromised their phone deliberately or Finding an ability to access other user's data from servers (i.e., data of another User and the data which not already exposed via APIs which as as per published features, ToS and Privacy policy). The ability to access aggregated, anonymized, randomized information is not a hack - its by design.

What is not a hack?

Decompiling, see one's own information and theability to access data on the phone when the phone has been deliberately compromisedby the user - unlocked or enabling ADB mode and exposing the IP address, etc. Theseare all expected behavior and require the user to go to extraordinary efforts to make theirown data visible - at which point, all their data may be compromised, not just the AarogyaSetu App.

Commonly flagged Issues by Code Analyzers or Scanners

1. **Missing Google Play Services Updated Security Provider(AndroidManifest.xml:30)**

    a) Android relies on a security Provider to provide secure network communications. However, from time to time, vulnerabilities are found in the default security provider. To protect against these vulnerabilities, Google Play

[1]*As on 27th May 2020, the FAQs will be revised from time to time*

services provides a way to automatically update a device's security provider to protect against known exploits. By calling Google Play services methods, your app can ensure that it's running on a device that has the latest updates to protect against known exploits.

For example, a vulnerability was discovered in OpenSSL (CVE-2014-0224) that can leave apps open to a "man-in-the-middle" attack that decrypts secure traffic without either side knowing.

b) To update a device's security provider, ProviderInstaller class is used. The installIfNeededAsync() method return normally if the device's Provider is successfully updated (or is already up-to-date) else throws exception.

installIfNeededAsync() is being used in app, so this mitigates the issue flagged above.

2. Missing Component Permission(AndroidManifest.xml:68,70,79,85), insecure Component receiver Issues

a) The "**exported**" attribute describes whether or not someone else can be allowed to use a particular activity. So, if you have "**exported**=false" on an Activity, no other app, or even the **Android** system itself, can launch it

b) android:exported is by default false for services if there is no filters. So in the case of NotificationRestoreService it can't be invoked externally.

Refer https://developer.android.com/guide/topics/manifest/service-element for more details

c) BootUpReceiver is intended to be invoked public by the system so it has ACTION_BOOT_COMPLETED as an intent filter.
Only the system can broadcast the filter "android.intent.action.ACTION_BOOT_COMPLETED" so there is no security threat here.

3. Unnecessary Permission(AndroidManifest.xml:9,10,11,12,13) and Android Network(AndroidManifest.xml:70,85)

a) The mentioned Permissions are required forthe app to function properly.
b) As the App is based on Bluetooth Contact Tracing, It is required to run foreground services like Bluetooth scanning service.

4. **Weak Encryption:Insecure, Insecure Randomness, ENCRYPTED_KEY_NAME, Inadequate RSA padding, Mode of Operation initCipherForLessThanM(),Weak Encryption Insecure mode of Operation**

    a) RSA ECB encryption mode is used in order to provide App compatibility on lower versions of Android.
    b) Degrading the cipher suite doesn't allow the attacker to get the user's data on the fly, as the data being stored locally on the device is anonymized and doesn't disclose the user's identity.

5. **Insecure Shared Preferences:-Location: SharedPref.java**
   Impact: Shared preferences is accessible through third party tools and sensitive information such as encryption key in this case can be extracted.

   Shared preference is local to the application but still if the phone is rooted the data stored in the shared preference can be extracted. However, AarogyaSetu App encrypts the data stored in shared preferences also. Also, the Users are advised not to use the App on a rooted phone.

6. **Encryption key is stored in shared preferences:-**
   Location: SharedPref.java
   Shared preference is local to the application but still if the phone is rooted the data stored in the shared preference can be extracted. However, AarogyaSetu App encrypts the data stored in shared preferences; the Keys are stored in Android key Store. The App does not store any sensitive data in Shared Preferences. Also, the Users are advised not to use the App on a rooted phone.

7. **Java script is enabled which can cause java script injection**
   Location: HomeActivity.java
   Java Script can only be invoked by owner and only within the app context.

8. **URL endpoints (API paths) are not encrypted and visible.**
   Data should be encrypted not endpoint. Encrypting endpoints will also lead to URL Decryption on each API call and might lead to draining more battery. In addition, Knowing the API endpoint is anyway very easy as each API call can be viewed by placing a proxy in between however, the same doesn't lead to any security issue.

9. **Cryptographic Vulnerability: Hardcoded Encryption Key or API Key**
   Hardcoded key is API key not encryption key. This API Key does not expose any sensitive data. This API key is used to interact with the backend for generating the OTP at the time of User Registration. The possible misuse of this API Key is very less,

as it just used for OTP generation. The User can try SMS bombing, but sufficient safeguard has been built-in by rate-limiting the OTP.

10. **Code Obfuscation: Encryption/Decryption Utility class is visible. Class names and DB queries are visible.**
    Since standard encryption and decryption method is used so nothing to hide in interface class.The Data on the phone is already encrypted.

11. **Some Activity/class of the App can be modified using Debugging (ADB) or other tools to load external content.**
    All classes and activities in the App are secured and will only load content which are allowed by the App.If the phone is not rooted and if the user is not using any debugging tools/emulator, then external content cannot be loaded. The App is not designed to run on rooted devices and majority of the Users don't have any debugging enabled on their phone. If the User intentionally tampers with the application, then it's being done it at their own risk and the same is true for any other App. Howeverdespite, doing all the tampering, the user won't be able to access personal data of other Aarogyasetu users.

12. **Disabled SSL CA Validation and Certificate Pinning**
    This is a false positive. SSL Pinning has already been implemented in the App and it is up to date.

13. **External Data in Raw SQL queries, this can potentially lead to a local SQL Injection**
    In general Applications use raw SQL queries for processing. No SQL Injection vulnerability exists in the database.

14. **Improper Error Handling.**
    No sensitive information is disclosed in the errors.

15. **The Acceptance to Terms and Conditions can be bypassed**
    This is normally done by by-passing the SSL Pinning and intercepting the request and modifying the request and response. This doesn't pose any security threat. Even if you by-pass the acceptance to terms, it doesn't change the Application's functions or features, nor does it disclose any sensitive data.

16. **Multiple HTTP Methods are enabled**
    The enabled HTTP Methods does not disclose any sensitive information

17. The root detection of App can be bypassed

The App doesn't allow authentication on a rooted device. However, it is possible to use any other third-party App to cloak the rooting. This cloaking of root can be done in general for all Apps running on the phone. This is not specific to Aarogya Setu. Hence, the Users are advised not to use the App on a rooted phone or use ADB or use any other 3rd party Apps which could bypass the Android security checks.

---------- End of Document ----------