Report on the privacy risks of COVID-19 software

December 2020

This report was commissioned by AWO and undertaken by external technical consultants.

Executive Summary

This report provides a technical analysis of seven smartphone applications deployed in six countries (Brazil, Colombia, India, Iran, Lebanon, and South Africa) to be used by citizens as part of their country's national response strategy to fight the COVID-19 pandemic. All are *official apps*, in the sense that they have been developed by (or with the approval of) competent organizational units in their respective governments, including health authorities.

The focus of this technical analysis is on the potential privacy risks of using such apps for citizens. Specifically, the report provides an assessment of the following elements: *(i)* the architecture used for contact tracing, if any; *(ii)* the presence of elements that can impact on users' privacy negatively (use of dangerous permissions, presence of software development kits (SDKs), potential dissemination of personal data, and consistency of observed behaviors and statements of the privacy policy); and *(iii)* components and design choices that can affect the security of the app, such as the presence of potentially harmful behaviors or an incorrect use of hosting and communication capabilities.

The main findings of this report are:

- All the examined apps implement a functionality for contact tracing / exposure notification. Two of them—Coronavirus SUS (Brazil) and COVID Alert SA (South Africa)—rely on the Google-Apple Exposure Notification (GAEN) API, which enforces some privacy guarantees. The remaining apps use centralized approaches that are potentially more privacy harmful, including custom implementations—Aarogya Setu (India), COVA (India), and Mask (Iran)—or based on Singapore's BlueTrace technology—CoronaApp (Colombia) and MA3AN (Lebanon). In addition to exposure notifications, all apps include additional functionality that ranges from services offering official COVID-19 information or directions to nearby health centers, to a variety of services based on personal data collection with unclear purposes.
- The five apps that do not rely on the GAEN API have access to geolocation information about users. This constitutes a serious privacy risk that, coupled with the collection of unique user/device identifiers and other personal data, could enable function creep (i.e., use of the collected data for purposes other than those for which they were collected in the first place). At least one app acknowledges in its privacy policy that collected data may be used for law-enforcement purposes.
- All seven examined apps integrate third-party software libraries with tracking and monitoring capabilities. The list includes well-known service providers such as Google's Firebase and Google Mobile Services, and also other tracking and analytics companies such as OneSignal, Sentry, and the Iran-focused Pushe.co. Even if the inclusion of such components does not automatically imply that their services are used for user tracking, their presence constitutes a risk and is bad privacy practice. None of the examined apps acknowledges in its privacy policy that the service providers process user data. One app—COVID Alert SA (South Africa)—integrate the WhatsApp SDK to facilitate contact with the National Department of Health through this platform. This entails potential privacy risks, disclosure of sensitive data to third-parties, and facilitates abuse.
- We find deficiencies in terms of transparency in all the analyzed apps, though there are significant differences across them. Only three of the examined apps have made their source code available for inspection. The majority make use of code obfuscation, reflection, and other anti-analysis techniques that make it difficult to reliably determine or confirm the presence of some behaviors. Such measures to hinder analysis negatively impact the trust in the apps and are misaligned with international transparency and privacy engineering recommendations.

The findings here reported constitute a description of technical elements found in the apps and their potential impact on users' privacy. They are not intended to be used as a legal or ethical assessment of the examined systems. Transparency and legal aspects (including, e.g., the completeness of privacy policies with respect to applicable regulatory provisions, the adequacy of consent forms, or questions related to data retention, sharing, and processing) are beyond the scope of this report.

Contents

1 COVID-19 CONTACT-TRACING APPS

On 3 April 2019, the World Health Organization (WHO) Digital Health Technical Advisory Group proposed the development of technical solutions to trace and control the development of the COVID-19 infection, known as **contact-tracing applications**.[1] Countries like Singapore,[2] Australia,[3] and South Korea[4] became early adopters in the deployment of contact-tracing applications—in the case of South Korea backed up by arguably privacy-intrusive methods that rely on linking massive databases such as geo-location or credit card transactions—to monitor the development of the pandemic and detect potential infections in a timely manner. The use of technology not only allows monitoring the spread of the virus at scale and almost in real-time, but also to notify citizens who may have been in proximity to infected people so that they can take precautions, such as self-isolating or remaining in quarantine. Other countries like Italy or Spain, as their health services were about to collapse, released applications offering self-diagnosis forms to assist citizens, along with official information and recommendations to remain safe in the context of the pandemic.

The high market penetration and ubiquitous nature of mobile devices, coupled with their ability to sense the environment using wireless interfaces like Bluetooth Low Energy (BLE), and obtain the user's location using A-GPS technologies, have made them an ideal platform for the development and fast deployment of contact-tracing software. There are four main ways to perform contact-tracing in a COVID-19 scenario:

- Mobile operator contact tracing using base-station level information provided by Mobile Network Operators (MNO) or other databases and systems with access to geo-location. The use of network-level data has limited accuracy because of the inherent lack of precision of the location information that is available.
- Location-based contact tracing. This method relies on mobile apps' ability to geo-locate users using A-GPS technologies and other location sensors such as network-based positioning technologies (e.g., using WiFi Access Points).
- Proximity-based contact tracing. This method leverages mobile apps' ability to sense neighboring devices using wireless networking interfaces like BLE.
- Hybrid solutions, which combine at least two of the previous methods.

The location-based and proximity-based contact tracing solutions have attracted more interest . In the case of proximity-based solutions, users exchange ephemeral identifiers with other users nearby, thus building a historical record of devices that have been in proximity (i.e., within BLE coverage). In both cases, apps need to keep Bluetooth active and run in the background (i.e., they are active even when the user is not interacting with the app) to discover other nearby users.

However, the use of mobile technologies to monitor the spread of the virus has raised numerous concerns from privacy advocates and regulators from all over the world, as it could allow the tracking of users' social interactions and activities.[5] The European Data Protection Board (EDPB) released guidelines on the privacy-preserving use of location data and contact-tracing tools in the context of the COVID-19 outbreak.[6] The EDPB report stresses the need to comply with EU's General Data Protection Regulation (GDPR) and the principles of privacy-by-design and privacy-by-default.

[1] https://www.who.int/news/item/03-04-2020-digital-technology-for-covid-19-response
[2] https://www.tracetogether.gov.sg/
[3] https://play.google.com/store/apps/details?id=au.gov.health.covidsafe
[4] https://wwwnc.cdc.gov/eid/article/26/10/20-1315_article
[5] https://privacyinternational.org/examples/location-data-and-covid-19
[6] https://edpb.europa.eu/our-work-tools/our-documents/ohjeet/guidelines-042020-use-location-data-and-contact-tracing-tools_en

Increasing regulatory and societal pressures demanded that privacy-preserving contact-tracing tools be implemented in Europe and other world regions. This pressure has triggered a race for different proximity-based contact-tracing architectures giving birth to two main paradigms: centralized and decentralized. The key difference between both approaches is how apps generate and manage the identifiers that will be later be exchanged with nearby users and, therefore, how they identify exposed users:

- In the centralized approach, the ephemeral identifiers are generated by a central authority, typically the national health service. If the user is diagnosed as COVID-19 positive, the user uploads to the central authority the list of identifiers sensed in the last T days, with $T = 15$ days being a usual value. The rest of the users check regularly with the servers to see whether they have been in proximity to a COVID-19 positive individual. Notable examples of implementations of centralized approaches are Singapore's TraceTogether and France's Robert[7] and Desire[8] architectures.
- In the decentralized architecture, with DP-3T being the first architecture design and prototype of this kind[9], the identifiers exchanged between devices are generated locally by the app running on end-user devices. When a user running the app is diagnosed as COVID-19 positive, the app sends the list of seen identifiers to the server, which are later retrieved by other devices to determine if they have been in their proximity.

The centralized approach gives, a priori, health authorities more control over the development of the pandemic, and has the potential to provide more fine-grained and timely data to epidemiologists. However, it causes more privacy harm to users than the decentralized approach by giving more control to a central authority. In fact, this could raise concerns if this technology is used in countries with a culture of excessive surveillance, and it could also enable function creep (e.g., user discrimination or inequality) by using the collected data for different purposes than originally planned.[10] [11]

Because of the potential privacy limitations in the centralized design, Google and Apple developed and released the (Google/Apple) Exposure Notification (GAEN) Application Programming Interface (API) following the principles of the decentralized architecture. The use of this API is restricted to certified national health authorities.[12] [13] Additionally, in order to prevent any privacy risk to end users, GAEN prohibits apps using the API from collecting geolocation data or even unique identifiers, while providing compatibility and interoperability between both mobile platforms. GAEN follows DP-3T's approach and uses BLE interfaces to discover neighboring devices and privacy-preserving cryptography.

The use of mobile apps for contact tracing is now a global trend. Many countries, however, do not follow the privacy standards defined by the EDPB, Google, and Apple. One consequence is that they may link contact information (which reveals part of a user's social graph) with personal data and even geolocation. This raises important privacy concerns. Within this context, this report aims to provide a systematic and comprehensive technical analysis of the privacy risks associated with

[7]https://github.com/ROBERT-proximity-tracing/documents
[8]https://hal.inria.fr/hal-02570382/document
[9]https://github.com/DP-3T
[10]https://arxiv.org/pdf/2007.02806.pdf
[11]https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Data%20Protection%20and%20Security.pdf
[12]https://www.google.com/covid19/exposurenotifications/
[13]https://developer.apple.com/documentation/exposurenotification

the use of seven contact-tracing applications from Brazil, Colombia, India (2), Iran, Lebanon, and South Africa.

Many academic studies have analyzed the privacy risks of contact-tracing applications. Douglas Leith and Stephen Farrell performed an exhaustive analysis of contact-tracing applications in late April [14] focusing on Singapore's OpenTrace app. They identified disturbing privacy risks associated with the use of Google's third-party tracking service Firebase, and the need to authenticate with a real phone number to use the service, thus de-anonymizing users. More recently, Joel Reardon and several researchers also performed a general overview of contact tracing technologies and their associated privacy risks.[15]

[14]https://www.scss.tcd.ie/Doug.Leith/pubs/opentrace_privacy.pdf
[15]https://arxiv.org/ftp/arxiv/papers/2007/2007.02806.pdf

2 REPORT OBJECTIVES

The objective of this report is to analyze the privacy risks of the following Covid-19 applications:

Application	Package Name	Country
CoronApp - Colombia	co.gov.ins.guardianes	Colombia
Coronavirus SUS	br.gov.datasus.guardioes	Brazil
COVID Alert SA	za.gov.health.covidconnect	South Africa
Ma3an base	com.tedmob.moph.tracer	Lebanon
Aarogya Setu	nic.goi.aarogyasetu	India
Cova	in.gov.punjab.cova	India
Mask	ir.covidapp.android	Iran

Specifically, we will perform an analysis along the following dimensions:

(1) **Architectural assessment of the Android apps:** We will investigate the contact-tracing model implemented by each Android application (i.e., distributed or centralized contact-tracing) as well as the underlying technologies implemented to discover neighboring devices (e.g., GAEN API, other third-party contact-tracing technology, or custom BLE scanning implementations).
(2) **Privacy assessment:** This section consists of the following assessments:
   - Permission analysis and assessment, including the use of custom permissions.
   - Presence of third-party SDKs and software components, providing a description of their purpose and the organization providing the service.
   - Assessment of potential personal data dissemination and the organizations collecting the data. We will focus on the detection of the dissemination of unique users identifiers (UUIDs), like the IMEI number or national IDs, and geo-location data.
   - Network flows and data dissemination to third party servers, including the data sent and the recipients where possible.
   - A brief assessment of the completeness of privacy policies from a technical perspective, including the analysis of potential inconsistencies between privacy policies and observed behavior to inform a proper legal assessment by expert partners.

(3) **Security assessment:** This section will focus on detecting potentially harmful behaviors (e.g., presence of malicious code or attempts to gain privileged system access according to external AV engines) and opaque behaviors (e.g., user fingerprinting in the case of websites). Additionally, we will register the organizations providing support on the cloud (e.g., hosting providers and their approximate geo-location) and the correct use of networking protocols (e.g., use of encrypted protocols like HTTPS to upload sensitive data).

For this analysis, we use existing and validated methodologies proposed by previous academic research efforts in the area of mobile privacy and security, particularly static and dynamic analysis tools.

Disclaimer: We would like to note that this report is a mere technical analysis of the apps grounded in scientific principles and it is not intended to be a legal and ethical assessment of each analyzed sample. Transparency and legal aspects such as the completeness of privacy policies according to any applicable regulatory provisions, data collection purposes, completeness and aptness of consent forms, and aspects related to data retention and data sharing are outside the scope of this report.

2.1 How to interpret the results

The document is structured as follows. Section 3 describes the details of the methodology followed, including how the analyses are conducted, the tools used and the limitations of each method. Then, Section 4 provides background and general information on the analyses, which help explain the technical details of the individual reports and provide an overall picture of the applications. Concretely, it describes the different contact-tracing implementations, like GAEN (Section 4.1), the personal and sensitive data types exposed in the apps (Section 4.2), the permission model in Android and the different permissions observed across the apps (Section 4.3), the third-party components that might be included in the applications (Section 4.4) and finally a description of networking considerations, including a diagram of the flows observed from the apps to external third-party hosts (Section 4.5).

Lastly, individual reports for each app are provided in separate Appendixes. Each of these reports is (generally) structured in 7 sections:

(1) **App metadata**. Provides an overview about the app, like the fingerprint, package name, developer, versions, etc.
(2) **App Overview**. Describes app features as reported by official documentation and also observations from the analysis, including development notes (*e.g.*, whether the app is open-source or not, or if its code is obfuscated).
(3) **Permission Analysis**. Describes the permissions requested and discusses those that might pose a privacy threat, including where in the app they are being used, and contextualizing the purpose for which they may have been requested.
(4) **Third-party SDKs and Libraries**. Lists the external libraries included, and discusses those that might be used for tracking purposes.
(5) **Hostnames and networking protocols**. Analyzes the network flows observed during dynamic analysis, whether these are secured or not with the use of TLS-level standards, and also the potential hostnames being contacted.
(6) **Personal data dissemination**. Describes the personal and identifiable data potentially leaked, and where those data are being sent.
(7) **Summary**. Provides a general summary of the key findings with regard to each app from the technical perspective, and contextualizes it with the privacy policy and official app documentation.

3 METHODOLOGY

The research community relies on two fundamental methodologies to conduct mobile application analysis from a privacy perspective: static analysis and dynamic analysis. In order to help the reader to contextualize the rest of this section and the results reported in this document, we provide a high-level description of each approach, including their limitations:

- **Static analysis.** Static software analysis is the analysis of computer software that is performed without actually executing programs, *i.e.*, by inspecting the code. In some cases, the source code of the apps is not available and so the compiled object code needs to be analyzed. This is the case for most apps studied in this report. Static analysis can be performed automatically by using purpose-built tools—which incorporate rules and heuristics designed to detect either specific behaviors or general features—, or manually by human analysis—a more time-demanding process depending on the size and complexity of the app to be studied. Unfortunately, static analysis presents several limitations as in the case of developers making use of code obfuscation —a technique for creating object code that is difficult for humans to understand—before releasing the app. Additionally, static analysis techniques can produce false positives, as many code paths and features might not be triggered in runtime (*i.e.*, dead or legacy code). This report makes use of both strategies to draw a more accurate and complete picture of the contact tracing apps being studied.

- **Dynamic analysis.** Dynamic program analysis is the analysis of computer software that is performed by executing programs on a real device or testing environment (e.g., an emulator or an instrumented sandbox). For dynamic program analysis to be effective, the target program must be executed with sufficient test inputs (or manual interaction) to trigger as many app features and behaviors as possible. While running, the program is monitored to identify how it behaves, including at the network traffic level. While dynamic analysis produces actual evidence of ann app's behavior, it can fail to identify many behaviors of interest that will then be left undiscovered. In the case of static analysis, developers can make use of anti-testing techniques to identify adversarial environments or techniques such as certificate pinning and certificate transparency[16] to protect the app against network observers who can perform traffic analysis.

It is known and accepted that the use of these methodologies might render inconclusive results due to the use of anti-analysis techniques by application developers (thus impeding reliable code analysis), or the use of techniques like certificate pinning (thus impeding traffic analysis). As a result, we combine both approaches in order to detect and report as many behaviors as possible without significantly increasing the number of false positives.

3.1 Ethical considerations

Given the sensitivity of the applications under scrutiny in this report, we decided to follow the highest ethical standards when conducting dynamic analysis tests. In particular:

- We do not stimulate the applications with data that could potentially compromise the integrity and data quality of the system (e.g., by registering and interacting as fake COVID-19 users).
- We do not attempt to register with applications that require user authentication, with or without fake credentials.

[16]https://en.wikipedia.org/wiki/Certificate_Transparency

- We do not perform any pentesting-like analysis to identify whether data is stored correctly on the servers or if it could be potentially exposed to malicious actors due to unprotected servers and poor development and security practices. These practices are indeed illegal in certain countries if they are not notified beforehand.

3.2 Tools

To help analyze Android applications, we combine the complementary insights offered by the following tools:

- Libradar.[17] LibRadar is an open source tool developed to detect the presence of third-party libraries on mobile apps. It also reports the name of the SDK and its purpose. While the developers claim that the tool is obfuscation-resilient, we identified numerous cases where the tool failed to report SDKs. This could be caused by apps making use of obfuscation techniques or because LibRadar's SDK fingerprints have not been updated. To overcome this limitation, we complement the results from LibRadar with manual code inspection.
- NinjaDroid.[18] This is an open-source tool that is based on Androguard's reverse engineering tool.[19]. NinjaDroid provides analysts with information extracted from the APK package such as metadata, found URLs and shell commands hard-coded in the apps, and file entries.
- Jadx.[20] This is an open-source tool for producing Java source code from Android apk files. It is used to decompile the application and perform code inspection.
- Apktool.[21] As in the case of Jadx, Apktool is a popular open-source tool to reverse engineer and analyze Android apps.
- Analysis sandbox. We use a mobile threat intelligence platform that allows apps to run in an instrumented execution environment and actual behaviors to be observed. It also provides valuable static analysis tools. The platform is hosted in a Central European country, so the observed behaviors are those that a user from that location might observe. There is a possibility that the observations collected when running the apps from this vantage point might be different from those that would have been gathered in a different country.
- Wireshark.[22] This is a free and open-source traffic interception tool and analyzer. It allows us to parse and visualize the network traffic and packets' content, both in real time as the traffic traverses the network interface, and to do offline analysis of traffic captured previously and stored in PCAP format. We use this tool to analyze the traffic exchanged between the applications and the different network servers as reported by the sandbox. We note that this content can be analyzed if the traffic is not encrypted.

We complement and enhance the observations gathered through these tools with custom analysis scripts and rules to process the data by automatic means, obtain actual evidence of apps' behavior and correlate the information gathered from various sources (*e.g.*, dynamic analysis and static analysis data)

[17]https://pypi.org/project/LibRadar/
[18]https://github.com/rovellipaolo/NinjaDroid
[19]https://androguard.readthedocs.io
[20]https://github.com/skylot/jadx
[21]https://ibotpeaches.github.io/Apktool/
[22]https://www.wireshark.org/

4 TERMINOLOGY, GLOSSARY AND TECHNICAL DESCRIPTIONS

4.1 Contact-tracing implementations

In this study, we identified the following contact-tracing technologies using different signals:

```
com.google.android.gms.exposurenotification.ACTION\_EXPOSURE\_STATE\_UPDATED
com.google.android.gms.exposurenotification.ACTION\_EXPOSURE\_NOT\_FOUND
```

Listing 1. Intents to be declared by applications leveraging Gogole's Exposure Notification API

- **Google-Apple Exposure Notification (GAEN).** *(Decentralized)* In order to use this contact-tracing capability offered by Google Mobile Services, developers must declare access to the intents listed in code Listing 1. As mentioned before, apps making use of this API are prohibited from collecting unique user identifiers and location data.
- **Bluetrace.**[23] *(Centralized)* This is a library for contact tracing that follows Singapore's centralized model. Developers wanting to integrate this library must request access to Bluetooth's interface, and are also able to collect geo-location data. Its presence in Contact-Tracing apps can be detected by inspecting the app's `AndroidManifest.xml` file. Code listing 2 lists the intents associated with this technology. Its presence can be also identified via code inspection.
- **Custom contact-tracing.** In many cases, we could not attribute the contact-tracing features implemented in the app to any well-known library. It might be possible that in these cases, application developers implement their own Bluetooth-based solutions.

We did not identify any application using DP-3T[24] (decentralized) open source-solution.

As we will discuss in each individual app report, many of the apps studied in this technical report implement features and offer services that go beyond those needed for contact tracing. In some cases, they are complementary services like self-diagnosing tools, real-time data, official recommendations and guidelines, or safe passes. We discuss each one on a case-by-case basis in the individual app reports.

```
      <receiver android:name="co.gov.and.coronapp.bluetrace.boot.StartOnBootReceiver">
          <intent-filter>
              <action android:name="android.intent.action.BOOT\_COMPLETED"/>
              <action android:name="android.intent.action.QUICKBOOT\_POWERON"/>
          </intent-filter>
       </receiver>
       <service android:exported="false"
           android:name="co.gov.and.coronapp.bluetrace.services.FCMService">
          <intent-filter>
              <action android:name="com.google.firebase.MESSAGING\_EVENT"/>
          </intent-filter>
       </service>
       <meta-data
android:name="com.google.firebase.messaging.default\_notification\_icon"
android:resource="@drawable/ic\_notification\_service"/>
       <meta-data
android:name="com.google.firebase.messaging.default\_notification\_color"
```

[23]https://www.bluetrace.io
[24]https://github.com/DP-3T/

```
android:resource="@color/notification\_tint"/>
       <activity android:launchMode="singleTask"
           android:name="co.gov.and.coronapp.bluetrace.permissions.RequestLocationPermission"
           android:theme="@style/Theme.AppCompat.Transparent.NoActionBar"/>
       <activity android:launchMode="singleTask"
           android:name="co.gov.and.coronapp.bluetrace.permissions.RequestBluetoothPermission"
           android:theme="@style/Theme.AppCompat.Transparent.NoActionBar"/>
       <service android:foregroundServiceType="0x00000008"
           android:name="co.gov.and.coronapp.bluetrace.services.BluetoothMonitoringService"/>
       <receiver
           android:name="co.gov.and.coronapp.bluetrace.receivers.UpgradeReceiver">
          <intent-filter>
             <action android:name="android.intent.action.MY\_PACKAGE\_REPLACED"/>
          </intent-filter>
       </receiver>
```

Listing 2. Sample of CoronApp Colombia's Android Manifest file indicating the use of Bluetrace, Firebase and Crashlytics.

4.2 Personal and sensitive data types

Personally identifiable information (PII) is defined as any data that can be used to identify a specific individual. This section provides a definition of sensitive data types as well as PII types identified during the inspection of the apps.

User and Device IDs. In this category, we group together any sensitive personal information that could identify the user or the device uniquely at a global scale. We identified the following data types during our analysis:

- **ANDROID_ID (AID).** This value is a semi-persistent 64-bit hexadecimal string that is unique to each combination of app-signing key, user, and device. As a result, it allows users to be uniquely identified within the app. However, in previous Android versions it was a unique ID per device that allowed users to identified across all their installed apps. [25] [26]. As a result, the privacy implications of requesting this ID depend on the Android operating system version running on user devices. This ID is not protected by any Android permission.
- **Name and surname.** Many contact tracing applications request users to register and introduce their name and family name, which corresponds to the owner of the phone or the user of the app. Requesting this data type could be considered privacy-intrusive, as it allows the user to be uniquely identified.
- **National ID number.** Some contact tracing applications require users to provide a valid national ID number to register and use the service. This value identifies a citizen uniquely at a regional, state or country-level.
- **Email.** This is a unique identifier that many contact-tracing apps require for using the service. In most cases, this value corresponds to the owner of the phone's email address.
- **Phone number.** This is the phone number associated with the SIM card that is installed on the phone. This value could be extracted programmatically by contact-tracing apps or could be requested using a text box in the user interface. This information is often requested to perform two-factor authentication.

[25]https://developer.android.com/reference/android/provider/Settings.Secure
[26]https://developer.android.com/training/articles/user-data-ids

Additionally, some apps collect hardware, platform and network identifiers that help fingerprint the device, such as the device model, the Android version or the mobile service provider.

Geolocation. Android allows application developers to programmatically retrieve the geolocation of a given device (and user) through the permission-protected Geolocation API (see the permission description in the next section).

Location information consists of either fine-grained GPS coordinates or other information—mainly associated with network infrastructure components like base stations or WiFi Access points—that would allow the recipient to infer the user's location (*e.g.*, SSID or BSSID of the WiFi AP or base station IDs). In fact, Android offers two geolocation sensors: (i) GPS (accurate or fine-grained), and (ii) network-based positioning (less accurate or coarse-grained). These sensors allow the platform to provide geo-location data regardless of the specific device context (i.e., whether the user is indoors without GPS coverage, or outdoors). Geo-location data is considered to be highly sensitive as it could reveal a user's habits, home address, workplace data, and even religious beliefs (i.e., place of worship). It is, as a result, a protected data type in most privacy regulations.

Bluetooth beacons. By definition, contact tracing applications need to exchange keys and IDSs with nearby devices. These IDs could be persistent in time (i.e., always associated with a given device, such as the Bluetooth MAC address) or dynamic (i.e., rotating). Depending on the contact-tracing architecture implemented by the app, these values could be shared with the other devices using Bluetooth Low Energy interfaces, or with the cloud servers through Wi-Fi or the mobile operator network. These IDs might not pose any privacy risk by themselves, but they require attention when applications also request unique user IDs or geo-location data.

Recent contacts. Some apps might request to manually enter a list of persons with whom the user has been recently in close contact, and also relatives. Collected data can include any item that can be used to identify the person, such as their name, email or phone number. The collection of such data can be considered privacy intrusive, since it reveals part of the user's social graph.

4.3 Android Permission Analysis

Smartphones store sensitive personal data about users, including their location, sensor data about their surroundings, and their social interactions. To reduce the privacy risks of giving apps unfettered access to sensitive user data, Android implements permission-based mechanisms to regulate how applications can access protected data and features.[27] App developers must declare their intent to access permission-protected resources (e.g., access to contacts, camera, or location data) in their applications by including them in their applications' `AndroidManifest.xml` file. Every Android app must have an AndroidManifest.xml file that describes essential aspects of the application, including package name, permissions, and hardware and software features that it requires, including (in some cases) third-party libraries.[28]

The Android Open Source Project (AOSP) defines a standard set of permissions that are supported by most Android devices. These permissions are categorized by Android's documentation as normal and signature permissions (granted at install-time) and dangerous (granted at runtime) by their potential privacy or security risk:[29]

- Normal permissions. According to Android's official documentation, normal "*permissions allow access to data and actions that extend beyond your app's sandbox. However, the data*

[27]https://developer.android.com/training/basics/permissions
[28]https://developer.android.com/guide/topics/manifest/manifest-intro
[29]https://developer.android.com/guide/topics/permissions/overview

and actions present very little risk to the user's privacy, and the operation of other apps." The Internet permission is an example of normal permission.
- Signature permissions. If the app declares a signature permission that another app has defined, and if the two apps are signed by the same certificate, then the system grants permission to the first app at install time. Otherwise, the first app cannot be granted permission.
- Dangerous permissions. Permissions in this category can give apps additional access to restricted data (e.g., unique user identifiers or geo-location), and to perform restricted actions that might affect the system and other apps. Developers must request access at runtime before accessing these restricted data or resources, and users must grant them access.

A remarkable feature of the Android permission system is its extensibility. The Android framework allows any app developer (or Android device manufacturer) to define `custom permissions`, which can be requested by other application developers. By defining custom permissions, an app can share its resources and capabilities with other apps. However, these permissions are not necessarily documented and their purpose can be difficult to assess. The reason why one app requests a custom permission is to access the service provided by the app declaring it. However, even in such cases it might be difficult to identify which type of resource the app declaring the permission is protecting.

The following subsections provide and document all the Android permissions identified in the applications, grouped into two categories: AOSP permissions and custom permissions. We discuss their purpose (if known) as well as their potential uses —legitimate or not— in contact tracing applications. Further details will be provided in each application's individual reports.

4.3.1 AOSP permissions. The table below depicts the normal, signature, and "dangerous" permissions—those requiring user approval—that the studied Android apps had requested.

Permission	**Protection Level**
`android.permission.INTERNET` • **General Usage / Purpose:** Allows applications to open network connections to communicate with other devices or online services over the Internet. • **Covid-19 Usage / Implications:** Uploading/downloading data to/from back-end servers supporting the service, and third-party components.	Normal
`android.permission.ACCESS_NETWORK_STATE` • **General Usage / Purpose:** Allows applications to access information about network interfaces (e.g., available mobile network, or WiFi AP). • **Covid-19 Usage / Implications:** Allows applications to know whether a network interface is available, as well as its status (e.g., radio signal). This could allow apps to know whether they have network access prior establishing a network communication.	Normal
`android.permission.ACCESS_WIFI_STATE` • **General Usage / Purpose:** Allows applications to access information about Wi-Fi networks. • **Covid-19 Usage / Implications:** Same usages and implications as ACCESS_NETWORK_STATE permission.	Normal
`android.permission.VIBRATE` • **General Usage / Purpose:** Allows access to the vibrator. • **Covid-19 Usage / Implications:** Notifying users.	Normal
`android.permission.FOREGROUND_SERVICE` • **General Usage / Purpose:** Allows a regular application to use `Service.startForeground`, which is needed to give application's services (processes running in the background) the ability to run with the same CPU scheduling as other application components running in the foreground (e.g., User-interface components).	Normal

• **Covid-19 Usage / Implications:** There is nothing particularly concerning from a mere privacy or security standpoint.	
`android.permission.BLUETOOTH` • **General Usage / Purpose:** Allows applications to discover and connect to paired Bluetooth devices. • **Covid-19 Usage / Implications:** This permission is critical for contact-tracing apps as it allows them to discover and communicate with other neighboring devices.	Normal
`android.permission.BLUETOOTH_ADMIN` • **General Usage / Purpose:** Allows applications to discover and pair with neighboring bluetooth devices. • **Covid-19 Usage / Implications:** Same implications as the BLUETOOTH permission.	Normal
`android.permission.WAKE_LOCK` • **General Usage / Purpose:** Allows applications to use PowerManager WakeLocks to keep processor from sleeping or screen from dimming. • **Covid-19 Usage / Implications:** Will allow applications to remain active in the background, which is needed to discover and communicate with other neighboring devices continuously. It might have power-draining implications if not used correctly.	Normal
`android.permission.CAMERA` • **General Usage / Purpose:** Allows access to the camera. • **Covid-19 Usage / Implications:** It might not be needed for implementing basic contact-tracing features. See reports on each individual app for a discussion of the uses of this permission across apps.	Dangerous
`android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS` • **General Usage / Purpose:** Permission that allows app developers to whitelist their application for battery optimization, so it is not subject to performance degradation. Android recommends a careful use of this permission due to the potential impact it might have on battery life. • **Covid-19 Usage / Implications:** Similar to the WAKE_LOCK permission.	Normal
`android.permission.RECEIVE_BOOT_COMPLETED` • **General Usage / Purpose:** This permission allows the application to be launched automatically on boot, without any user interaction. • **Covid-19 Usage / Implications:** Allows contact-tracing to run as soon as the device boots up.	Normal
`android.permission.CALL_PHONE` • **General Usage / Purpose:** Allows an application to initiate a phone call without going through the *Dialer's* user interface for the user to confirm the call. • **Covid-19 Usage / Implications:** The reason for this permission depends on the features implemented by each individual app. This permission, as in the case of the Colombian app, allows users to reach out directly to the health authorities from the app. See reports on each individual app for a discussion of the uses of this permission across apps.	Dangerous
`android.permission.READ_EXTERNAL_STORAGE` • **General Usage / Purpose:** Allows an application to read from external storage (e.g., SDCard). It allows information to be read from shared storage. It can have potential security and privacy implications as apps holding this permission can have access to sensitive user data stored on the device (e.g., pictures or documents). • **Covid-19 Usage / Implications:** Access to this permission is not needed to implement contact-tracing applications. See reports on each individual app for a discussion of the uses of this permission across apps.	Dangerous
`android.permission.WRITE_EXTERNAL_STORAGE` • **General Usage / Purpose:** Allows an application to write to external storage. • **Covid-19 Usage / Implications:** Access to this permission is not needed to implement contact-tracing applications. See reports on each individual app for a discussion of the uses of this permission across apps.	Dangerous
`android.permission.ACCESS_FINE_LOCATION` • **General Usage / Purpose:** According to Android's official documentation, this permission allows an app to access the user's location using GPS.	Dangerous

• **Covid-19 Usage / Implications:** There are concerning privacy implications, as it can allow applications not only to know who a given user has been in close proximity with, but also where. The use of geo-location data in contact-tracing applications has been widely discouraged by many reports and development guidelines due to their intrusive nature. Apps using Google's and Apple's Exposure Notification API are prohibited from collecting location data. See reports on each individual app for a discussion of the uses of this permission across apps.	
`android.permission.ACCESS_COARSE_LOCATION` • **General Usage / Purpose:** According to Android's official documentation, this permission allows an app to access approximate location (i.e., through network positioning). • **Covid-19 Usage / Implications:** The same privacy implications as the ACCESS_FINE_LOCATION permission. See reports on each individual app for a discussion of the uses of this permission across apps.	Dangerous
`android.permission.ACCESS_BACKGROUND_LOCATION` • **General Usage / Purpose:** Allows applications to access location in the background (*i.e.*, even when the users are not interacting with the app) • **Covid-19 Usage / Implications:** The same privacy implications as the ACCESS_FINE_LOCATION permission. See reports on each individual app for a discussion of the uses of this permission across apps.	Dangerous

Table 2 shows the dangerous AOSP permissions requested by the analyzed apps.

Permission	CoronaApp	Cornavirus SUS	COVID Alert SA	Ma3an	Aarogya	Cova	Mask
`android.permission.CALL_PHONE`	●						
`android.permission.READ_EXTERNAL_STORAGE`							○
`android.permission.WRITE_EXTERNAL_STORAGE`							○
`android.permission.ACCESS_FINE_LOCATION`	●			●	●	●	○
`android.permission.ACCESS_COARSE_LOCATION`	●				●	●	○
`android.permission.ACCESS_BACKGROUND_LOCATION`					●	●	○
`android.permission.ACCESS_CAMERA`					●	●	●

Table 2. Dangerous AOSP permissions requested by the analyzed apps. The symbol ○ for Iran's Mask app denotes the fact those permissions were requested by many previous versions of the app except the one analyzed in this report.

4.3.2 Custom Permissions. This section presents the number of custom permissions identified in the applications studied. The majority of the custom permissions found are related to the implementation of "badge numbers", which are small numeric values that appear over an app icon to report the number of notifications within the app. While this feature is popular among iOS app developers, Android did not support this feature by default until Android 8. As a result, many Android device manufacturers implemented their own version. [30] These permissions do not necessarily pose any privacy risks to the end user but they signal the developers' intention to support users running old Android versions.

Permission	Protection Level
`android.permission.READ_APP_BADGE` • **General Usage / Purpose:** Custom permission enabled by Zuk mobile (this brand was a subsidiary of Lenovo and it ceased operations in 2017) to add iOS-like numeric notifications on top of Android app icons. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.anddoes.launcher.permission.UPDATE_COUNT` • **General Usage / Purpose:** Same as READ_APP_BADGE for Apex devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.htc.launcher.permission.READ_SETTINGS` • **General Usage / Purpose:** Same as READ_APP_BADGE for HTC devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.htc.launcher.permission.UPDATE_SHORTCUT` • **General Usage / Purpose:** Same as READ_APP_BADGE for HTC devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.huawei.android.launcher.permission.CHANGE_BADGE` • **General Usage / Purpose:** Same as READ_APP_BADGE for Huawei devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.huawei.android.launcher.permission.READ_SETTINGS` • **General Usage / Purpose:** Same as READ_APP_BADGE for Huawei devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.huawei.android.launcher.permission.WRITE_SETTINGS` • **General Usage / Purpose:** Same as READ_APP_BADGE for Huawei devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.majeur.launcher.permission.UPDATE_BADGE` • **General Usage / Purpose:** Same as READ_APP_BADGE for Solid devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.oppo.launcher.permission.READ_SETTINGS` • **General Usage / Purpose:** Same as READ_APP_BADGE for Oppo devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.oppo.launcher.permission.WRITE_SETTINGS` • **General Usage / Purpose:** Same as READ_APP_BADGE for Oppo devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.sec.android.provider.badge.permission.READ` • **General Usage / Purpose:** Same as READ_APP_BADGE for Samsung devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.sec.android.provider.badge.permission.WRITE` • **General Usage / Purpose:** Same as READ_APP_BADGE for Samsung devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.sonyericsson.home.permission.BROADCAST_BADGE` • **General Usage / Purpose:** Same as READ_APP_BADGE for Sony devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`com.sonymobile.home.permission.PROVIDER_INSERT_BADGE` • **General Usage / Purpose:** Same as READ_APP_BADGE for Sony devices. • **Covid-19 Usage / Implications:** Usability.	Normal
`me.everything.badger.permission.BADGE_COUNT_READ` • **General Usage / Purpose:** Same as READ_APP_BADGE for EverythingMe devices (brand is now defunt).	Normal

[30] https://distriqt.github.io/ANE-Notifications/u.Set%20Badge%20Number

• **Covid-19 Usage / Implications:** Usability.	
`me.everything.badger.permission.BADGE_COUNT_WRITE` • **General Usage / Purpose:** Same as READ_APP_BADGE for EverythingMe devices (brand is now defunt) • **Covid-19 Usage / Implications:** Usability.	Normal
`com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE` • **General Usage / Purpose:** According to the Google's Firebase documentation, this custom permission is needed to integrate its SDK in apps. Specifically, it is required to report events to the server, such as installation events or scheduling tasks, possibly for analytics purposes. • **Covid-19 Usage / Implications:** The presence of this permission in apps' AndroidManifest reveals the use of Google's third-party service on these apps, which might have potential privacy implications.	Normal
`com.tedmob.moph.tracer.permission.C2D_MESSAGE` • **General Usage / Purpose:** Push notifications (i.e., sending information from the server to the user) • **Covid-19 Usage / Implications:** Sending notifications to end users	Normal
`com.google.android.c2dm.permission.RECEIVE` • **General Usage / Purpose:** This permission is required by apps using Google's push notification service (*i.e.*, server-to-client notifications and communications). • **Covid-19 Usage / Implications:** Opens the door to potential eavesdropping attacks.	Normal

4.4 Third-party components

As with the web and desktop software, most app developers rely on third-party components —*e.g.*, libraries or software development kits (SDKs)—that they integrate into their apps to add particular desired functionalities. Such libraries represent the benefit of being modular, and being reusable (typical of object-oriented design), both of which are reflected in best software engineering practices.

These third-party libraries can speed up the development process or monetizing software (depending on the nature of the app) by providing a variety of different features, such as development support, bug reporting, cryptographic support, analytics, advertising, and graphics support. These development practices, however, may come at a privacy cost to end users by allowing third parties to monitor users' habits thanks to their ability to be present in millions of apps and websites. In the case of mobile applications, this is a concern as current mobile permission models—both for iOS and Android platforms—allow third-party SDKs to run in the same context as the app itself. In other words, SDKs embedded in a mobile application can access the same files, and permission-protected resources that the host app can access.

We classify the third-party libraries found in the applications under scrutiny by their purpose or function. First, we decided to perform a classification based on LibRadar's categorization; however, we noticed that it has some inconsistencies, so we enhanced it using information available on the web, either from the website of the product, from its source-code or from its documentation.

We identify the following two main categories of mobile SDKs: (1) development support SDKs, and (2) tracking / analytics SDKs. We discuss each case below.

Development Aid Tools. This category covers third-party libraries or SDKs whose purpose is to support or ease the development of the software, or integrate features to an application's code such as widgets, graphics, JSON and XML parsers.

Development support libraries are likely to be found in any mobile application, and—provided that they do not include malware code— they pose little or no risk to the user's privacy since the majority of them do not collect personal data. As a result, most of these SDKs do not need to be included in the privacy policy.

- Google Core Libraries for Java 6+ (also known a Guava).[31] This is a Google-maintained open-source library that offers several optimized capabilities for managing data structures, string manipulation, graphics, computation, cryptographic, and input/output (I/O). It is very popular amongst many Android apps, and there are multiple versions. It should cause no privacy harm.
- Android Support Libraries.[32] This is a Google-maintained library that allows Android app developers to support multiple API / Operating System versions, and also features that are not part of the Android standard APIs that can significantly ease app development. It is very popular amongst many Android apps and it should cause no privacy harm.
- AndroidX.[33] AndroidX is the new and enhanced version of Android Support Libraries. Google provides it to app developers.
- Kotlin/KotlinX.[34] Kotlin is a new cross-platform programming language that is fully supported by Android. These libraries expose a common standard API for multiplatform projects. They assist developing software in this language.
- Google Protocol Buffers.[35] Protocol buffers are a new serialization mechanism developed by Google for efficient data management and communications. This library allows developers to integrate these capabilities in their software. It is platform-independent and can be easily extensible by the developers. The presence of this library should cause no privacy damage to end-users.
- Google Gson:[36] This is a Google-maintained open-source library that allows developing to convert (or serialize) Java Objects into a JSON representation, and vice-versa. The presence of this library should cause no privacy damage to end-users.
- FasterXML.[37], [38] This is an open-source library used to manipulate and parse XML and JSON files for Java. It offers capabilities similar to those of Google Gson. The presence of this library should cause no privacy damage to end-users.
- Glide:[39] Glide is a fast and efficient open-source media management and image loading library for Android. This library wraps media decoding, memory and disk caching, and resource pooling into a simple and easy to use interface. It was developed by Bump Technologies, which was acquired by Google in 2013. The presence of this library should cause no privacy damage to end-users.
- JavaX Annotation API.[40] [41] Open-source library that allows integrating common annotations in Java code (i.e., mark source code that has been generated or to distinguish between user-written code and generated code). The presence of this library should cause no privacy damage to end-users.

[31]https://github.com/google/guava
[32]https://developer.android.com/topic/libraries/support-library
[33]https://developer.android.com/jetpack/androidx
[34]https://ktor.io/docs/kotlinx-index.html
[35]https://developers.google.com/protocol-buffers
[36]https://github.com/google/gson
[37]http://fasterxml.com/
[38]https://github.com/FasterXML
[39]https://github.com/bumptech/glide
[40]https://mvnrepository.com/artifact/javax.annotation/javax.annotation-api
[41]https://docs.oracle.com/javase/7/docs/api/javax/annotation/package-summary.html

- MaterialProgressBar.[42] Open-source library developed by Hai Zhang for Material Design. The presence of this library should cause no privacy damage to end-users.
- Joda Time.[43] Open-source library for date and time operations that overcomes some of the limitations of Java's default one. The presence of this library should cause no privacy damage to end-users.
- Threeteen.[44] Open-source date and time library (same purposes as Joda Time). This library should not cause any direct privacy risk to users.
- Zebra Xing.[45] ZXing ("zebra crossing") is an open-source Java multi-format 1D/2D barcode image processing library, including QR codes. It can be used to generate and parse QR codes (e.g. in the case of CoronaApp from Colombia, users can use QR codes generated by the app as a safe passage. The presence of this library per-se should cause no privacy damage to end-users, and it depends on how it is used.
- OKHTTP 3.0. [46] This library offers support for implementing network protocols such as HTTP, and HTTPS without having to depend on those offered by the Android framework. Networking SDKs might not collect sensitive and behavioral data from users. Babylon Health - Certificate Transparency for Android.[47] This is an open-source library developed by
- BabylonHealth[48] to protect apps from man-in-the-middle (MITM) attacks. Its aim is to overcome the limitations of certificate pinning by making use of the Certificate Transparency initiative.[49]. This library, therefore, enhances applications' network security guarantees and does not cause any privacy harm to end-users.
- Apache Commons.[50] The Apache Commons open-source project combines multiple capabilities and features for development support, from geometry and math libraries to XML parsing and dynamic proxy creation. This SDK might not collect sensitive and behavioral data from users.
- Apache Cordova.[51] This is an open-source mobile development framework that allows application developers to create mobile applications using web technologies (e.g., HTML5, CSS3, and JavaScript) and therefore, ease cross-platform development. This SDK might not collect sensitive and behavioral data from users.
- ReactiveX.[52] ReactiveX is an open-source multi-language library for building asynchronous and event-based programs. This project is a collection of several open-source programs. This SDK might not collect sensitive and behavioral data from users.
- Material Dialogs.[53] Open-source library to implement UI dialogs in Android. It supports both Java and Kotlin. This SDK does not collect sensitive or behavioral data from users.

[42]https://github.com/DreaminginCodeZH/MaterialProgressBar
[43]https://www.joda.org/joda-time/
[44]https://www.threeten.org/
[45]https://github.com/zxing/zxing
[46]https://square.github.io/okhttp/
[47]https://github.com/babylonhealth/certificate-transparency-android
[48]https://www.babylonhealth.com/
[49]https://www.certificate-transparency.org/
[50]https://commons.apache.org/
[51]https://cordova.apache.org/
[52]http://reactivex.io/
[53]https://github.com/afollestad/material-dialogs

- AirBnB Lottie.[54] This open-source library is implemented and maintained by AirBnB.[55] It allows the user to render natively visual effects based on Adobe After Effects on Android, iOS and React Native applications. It should not collect any sensitive user data.
- RxBindings. [56] An open-source library developed by Jake Wharton. It offers RxJava binding APIs for Android UI widgets. This SDK does not collect sensitive and behavioral data from users.
- Parceler.[57] Efficient java object serialization library to share data between contexts. This SDK might not collect sensitive and behavioral data from users.
- Michael Rocks Libphonenumber.[58] An open-source Android port of Google's Libphonenumber library to parse phone numbers. This SDK might not collect sensitive and behavioral data from users.
- CapacitorJS.[59] Capacitor JS is, as in the case of Apache Cordova, a runtime engine for running cross-platform applications (similar to Native React) implemented using web technologies (e.g., JavaScript, HTML and CSS). The presence of this SDK should not cause any privacy harm.
- Dagger.[60] This is an open-source development tool that allows inject dependencies in compilation time. It is maintained by Google.[61] The presence of this library should not cause any privacy harm.
- LiteGlue.[62] LiteGlue is a collection of open-source libraries for SQLite (databases). This library should not cause any direct privacy risk to users.
- Leolin ShortcutBadger.[63] This library implements notifications on Android app icons. This library should not cause any direct privacy risk to users.
- Adobe Phonegap Push.[64] Phonegap is a proprietary push notification service developed by Adobe. It also allows the building of mobile applications using web technologies (it is an open-source distribution of Apache Cordova). We could not determine the privacy risks of this service and the implications of delegating push notifications related to covid-19 through Adobe.
- Calligraphy.[65] Developed by Christopher Jenkins, this is a library that allows developers to use custom fonts in Android. This library should not cause any direct privacy risk to users.
- PinView.[66] This open-source library provides a widget that allows users to enter a PIN, one-time password (OTP) or a password on the app. This library should not cause any direct privacy risk to users.
- EventBus.[67] This open-source library provides a publish/subscribe event bus for Android and Java that is used to simplify the communication among components. This library should not cause any direct privacy risk to users.

[54] https://airbnb.io/lottie/
[55] https://github.com/airbnb/lottie
[56] https://github.com/JakeWharton/RxBinding
[57] http://parceler.org/
[58] https://github.com/MichaelRocks/libphonenumber-android
[59] https://capacitorjs.com/
[60] https://dagger.dev/
[61] https://github.com/google/dagger
[62] https://github.com/liteglue
[63] https://github.com/leolin310148/ShortcutBadger
[64] http://docs.phonegap.com/tutorials/develop/push-notifications/
[65] https://github.com/chrisjenx/Calligraphy
[66] https://github.com/mukeshsolanki/android-otpview-pinview
[67] https://github.com/greenrobot/EventBus

- Moshi.[68] This is an open-source JSON manipulation library for Android and Java. This library should not cause any direct privacy risk to users.
- PersianMaterialDateTimePicker.[69] This library provides apps with a hijri/shamsi (Iran Calendar) date picker and a regular time picker. This library should not cause any direct privacy risk to users.
- ZXing Android Embedded.[70] This provides a barcode scanning library for Android based on the popular Zebra Xing library for decoding (see above). Camera permission is required to perform the action. While this library should not cause any direct privacy risk to users, its misuse could.
- MPAndroidChart.[71] A chart plotting library for Android. This library should not cause any direct privacy risk to users.
- Neshan.[72] This SDK provides apps with map-based navigation services for Iranian cities, including street-level traffic data, location services, and user mobility data.
- Android-SpinKit .[73] This library is provided by an independent software engineer from India, and is used to select images from the gallery or to capture images using the camera.
- Dhaval2404 ImagePicker.[74] This library is used for loading various animations and GIFs in Android.
- CodeScanner.[75] This library provides functionality to scan various formats of codes, like QR or Aztec (therefore the library requires the Camera permission). This library is used in the analyzed apps to scan QR or other electronic codes.
- Silicompressor.[76] This is a library developed by a Russian software developer and provides functionality for Video and Image compression in Android.
- Nabinbhandari Android-Permissions.[77] This is a library used to handle runtime permissions in Android. It is used to check for the appropriate permissions whenever these are needed.
- Soffritti YoutubePlayer.[78] This is a library developed by a Google engineer from London, UK, and used to provide a custom YouTube Player that can be embedded in the different components of the applications.
- RootBeer.[79] This is a popular library used to check whether a device is *rooted* or not. It relies on a native library (tool-checker) for checking for the su binary in the system.
- FilePicker.[80] This is a lightweight Android file picker library created by a Chinese developer, which helps to navigate, select and pick images from the devices' file systems.
- uCrop.[81] This is a library used for image cropping, developed by a Ukrainian developer.

We acknowledge that the apps under scrutiny might integrate other unpopular and open-source libraries that had remained undetected.

[68]https://github.com/square/moshi
[69]https://github.com/mohamad-amin/PersianMaterialDateTimePicker
[70]https://github.com/journeyapps/zxing-android-embedded
[71]https://github.com/PhilJay/MPAndroidChart
[72]https://developers.neshan.org
[73]https://github.com/Dhaval2404/ImagePicker
[74]https://github.com/ybq/Android-SpinKit
[75]https://github.com/yuriy-budiyev/code-scanner
[76]https://github.com/Tourenathan-G5organisation/SiliCompressor
[77]https://github.com/nabinbhandari/Android-Permissions
[78]https://github.com/PierfrancescoSoffritti/android-youtube-player
[79]https://github.com/scottyab/rootbeer
[80]https://github.com/fishwjy/MultiType-FilePicker
[81]https://github.com/Yalantis/uCrop

SDKs with tracking and monitoring capabilities. Many companies like Google provide analytics tools to allow developers to understand how users interact with the application, detect bugs and errors, measure user adoption and retention, measure audiences, optimize usability, or even to detect fraudulent usages of the app. Many of these services also offer development support and other capabilities and features to assist application developers in the development, deployment and release stages of their apps.

Across the Covid-19 apps studied, we identified several SDKs within this category:

- Google Firebase. [82] Firebase is one of the most popular and powerful SDKs in this category. This library combines multiple analytics and development support services including analytics (via Google Analytics),[83] secure user authentication,[84], crash reporting (via a service previously known as Crashlytics),[85] in-app messaging[86], performance monitoring,[87] and A/B testing[88]. Some of these products and capabilities integrated in Firebase like A/B testing, Analytics, App Distribution, Crashlytics, Google Cloud Messaging, In-app Messaging, Performance Monitoring or App Indexing are offered free of charge by Google to app developers. The others are offered for a flat rate or on a pay-as-you-go basis.[89] Firebase also eases integration of other Google products like Google Ads, AdMob (for advertisements), or Google Marketing Platform. Many research efforts have found that analytics and crash reporting services are prone to collecting usage telemetry and personal data to deliver their service, and for secondary usages like advertisements. For this reason, Firebase might act as a "data controller" when integrated in many apps under the requirements of the GDPR and CCPA [90]: *"When customers use Firebase, Google is generally a data processor under GDPR and processes personal data on their behalf. Similarly, when customers use Firebase, Google generally operates as a service provider under the CCPA handling personal information on their behalf. "* In the case of Crashlytics, which has its own Terms of Service (ToS),[91] it states: *"At all times during the term of this Agreement, Developer shall maintain a privacy policy: (a) that is readily accessible to users from its website or within its online service (as applicable), (b) that fully and accurately discloses to its users what information is collected about its users, and (c) that states that such information is disclosed to and processed by third party providers like Google in the manner contemplated by the Services".* This wording suggests that when this SDK is used in Covid-19 apps, their respective privacy policy should mention their presence. We will investigate the correct use of this library in each individual app report when applicable.
- OneSignal.[92] OneSignal is a push notification service. It offers free and premium push notification services. According to its privacy policy, it records "*what push notifications and end user has been sent*", "*precise location information*", "*mobile advertising identifiers*", as well as "*a unique cookie identifier, which may uniquely identify an end users*".[93] Therefore, it might have severe privacy implications. The privacy policy also indicates that this information can be used to *"perform any of the above functions, or other marketing or analytics services. Or,*

[82]https://firebase.google.com/
[83]https://firebase.google.com/products/analytics
[84]https://firebase.google.com/products/auth
[85]https://firebase.google.com/products/crashlytics
[86]https://firebase.google.com/products/in-app-messaging
[87]https://firebase.google.com/products/performance
[88]https://firebase.google.com/products/ab-testing
[89]https://firebase.google.com/pricing
[90]https://firebase.google.com/support/privacy
[91]https://firebase.google.com/terms/crashlytics
[92]https://onesignal.com/
[93]https://onesignal.com/privacy_policy

we may aggregate and create data models to do this – creating algorithms in order to predict certain trends and things that different End Users might have in common, for instance.”

- Fabric.[94] This SDK was an extension of Crashlytics released in 2014—back in the day, Crashlytics was owned by Twitter before being sold to Google—that offered services like mobile app analytics, beta distribution, and user identity and authentication, among many other features.[95] Now, this product is defunct and it is fully integrated in Firebase. Their presence in apps suggests poor development practices as it integrates a component that has not been maintained for many years. It presents similar privacy risks to the ones previously reported for Firebase.
- Google Mobile Services.[96] Google Mobile Services is a collection of applications and APIs that allow users to have access to Google’s most popular mobile apps (e.g., Gmail, Google Maps, Google Play, Search, Chrome, Youtube, etc.) and APIs to integrate these services in mobile applications. Apps integrating Google Mobile Service APIs can access to a variety of services including analytics, advertising, cloud integration, or geo-location.[97] In normal circumstances, the use of features implemented by this SDK could pose potential privacy risks to users, similar to those related to Firebase. However, in the case of contact-tracing applications, its use is justified as it provides the capabilities implemented by Google’s Exposure Notification API.[98]
- Pushe.co.[99] This is a Tehran-based company providing push notification and app analytics services for mobile and web services. The official websites states that the service is already used by 6K+ developers in Android apps with over 1B installations. Similar to other marketing-oriented analytics SDKs like Google Analytics, Pushe.co allows developers to track users, classify them according to different behavioral traits, and target them with custom push notification. Specifically, the Android SDK provides developers the ability to track users by geographical location, phone model and brand, user operator, internet type, rate, gender, age, and custom tags.[100] The privacy policy[101] lists the items collected from users, which include a hardware fingerprint of the device (which is a persistent identifier), a number of unique identifiers (Android ID, Google Ad ID, SIM card data), the lists of apps installed on the device, location, network-related data (IP address, volume of traffic exchanged, MAC, the names of Wi-Fi networks and their strength, etc.) and performance-related metrics. The privacy policy also states that this information can be shared with third parties. As in other cases described above, the presence of this SDK in COVID-19 apps may have severe privacy implications.
- Sentry.[102] Sentry is an SDK—it supports 87 different platforms— for tracking mobile installations, crash analytics, and error reporting similar to the capabilities offered by Crashlytics. It can be used to track user adoption, use of the app and its features, and to identify crashes and potential bugs. According to its documentation, developers can track users in different ways: from IDs to email addresses and at the IP-level.[103] Sentry also allows developers to define custom tags or events. If misused, this capability could entail severe data leakage (i.e.,

[94] https://get.fabric.io/
[95] https://www.wired.com/2014/10/twitter-fabric-sdk/
[96] https://www.android.com/gms/
[97] https://developers.google.com/android
[98] https://developers.google.com/android/exposure-notifications/exposure-notifications-api
[99] https://pushe.co/
[100] https://pushe.co/product
[101] https://pushe.co/privacy
[102] https://sentry.io/for/mobile/
[103] https://docs.sentry.io/platforms/android/enriching-events/identify-user/

SDK	CoronaApp	Cornavirus SUS	COVID Alert SA	Ma3an	Aarogya	Cova	Mask
Google Firebase	●	●	●	●			●
Google Mobile Services	●	●	●	●	●	●	●
Google Crashlytics					●		
OneSignal				●			
Fabric	●			●	●		
Pushe.co							●
Sentry							●

Table 4. SDKs with tracking and monitoring capabilities found in the analyzed apps.

if developers report to Sentry's servers a tag "covid19_positive"). In fact, Sentry's documentation suggests that developers should be careful and ensure that sensitive data never reaches Sentry servers.[104]

Additionally, Section 4.1 provides more information about third-party SDKs and libraries specialized in offering contact-tracing technologies to app developers.

Table 4 shows the presence of SDKs with tracking and monitoring capabilities in the analyzed apps.

4.5 Networking considerations

Most Android applications communicate with external servers using various network protocols. Typically, these servers are offered as web services, e.g., by means of Application Programming Interfaces (APIs) that allow the application to automatically interact with the server and use services it offers.

The most common networking protocols in mobile applications are HTTP and HTTPS. The difference between them is important, since the former sends the data in cleartext—i.e., without protecting its confidentiality—while the latter encrypts the communications. The use of HTTP has profound security and privacy implications, since anyone monitoring the communication between the client and the server (e.g. an Internet Service Provider) can easily eavesdrop on the information. Additionally, server authentication is not guaranteed. HTTPS protocol, instead, encrypts the communication between the client and the server, and also the server is authenticated by means of a digital certificate. This certificate provides information about the organization hosting the service, and thus allows who the applications are communicating with to be understood with certain confidence.

For each of the apps, we analyze what the contacted servers are, and which protocols are used for this communication. Concretely, we will report the following:

- Hostname. The name of the domain being contacted, which in the absence of further information, might be the only way to know who owns the server.

[104]https://docs.sentry.io/platforms/android/data-management/sensitive-data/

- IP and hosting provider. We resolve the domain to get the actual IP address. Then, by leveraging the `whois` protocol, we obtain information about the Autonomous System (AS), which allows us to identify the network operator or Internet Service Provider for the IP.
- If the connection is made by means of the TLS protocol, we report information about the certificate owner from the server.

We analyze the network flows using dynamic analysis. Also, we rely on static analysis to look for hostnames. In the latter, we confirm that these are potentially contacted by analyzing the context where they are found. Figure 1 shows the hosts that have been contacted by more than one application across the apps. We note that this diagram considers those hostnames detected either using dynamic analysis, or observed by static analysis.

We exclude those observed by static analysis if they are found on all the applications to minimize false positives. The reason for this is that the hostnames are likely present due to common functionality of every Android app (*e.g.*, the host `schemas.android.com`) or services and capabilities offered by embedded third-party SDKs and libraries. To ease visualization, we also exclude the host `www.googleapis.com`, which is contacted multiple times by all the apps except the Iranian one.

We also report the information in Table 5. We observe domains from Google (*e.g.*, `play.google.com` or `www.googleadservices.com`) and from popular libraries (*e.g.*, `app-measurement.com`, `crashlytics.com` or `firebase.google.com`). We also observe hosts related to other libraries, like `github.com` or `www.apache.org`. During the analyses done for each individual apk, we provide further details for these hosts and discuss the implications of the different apps contacting them.

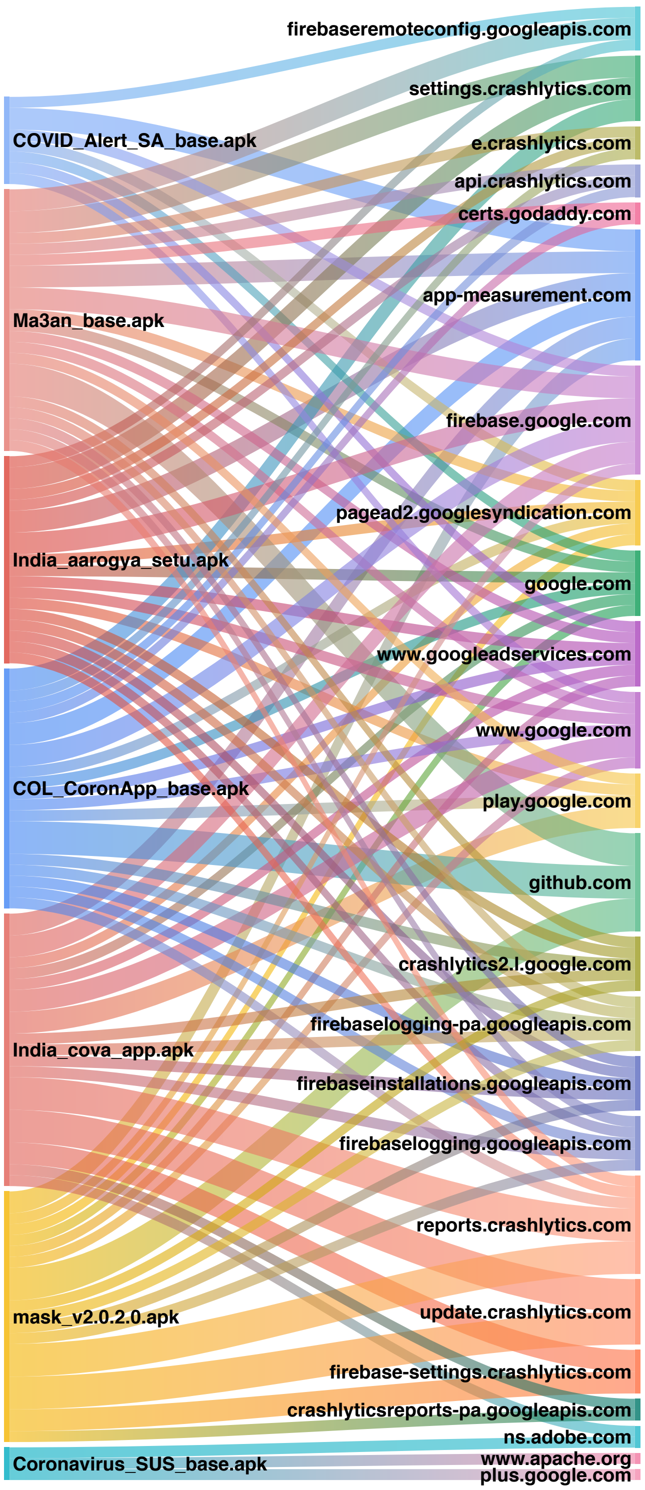


Fig. 1. Hosts contacted by more than one app, excluding those contacted by all the apps

Hostname	CoronApp	Coronavirus SUS	Aarogya	Cova	Ma3an	COVID Alert SA	Mask
app-measurement.com	●		●	●	●	●	●
pagead2.googlesyndication.com	●		●	●	●	●	●
google.com	●		●	●	●	●	●
www.googleadservices.com	●		●	●	●	●	●
www.google.com	●		●	●	●	●	●
firebase.google.com	●		●	●	●	●	●
reports.crashlytics.com	●		●	●	●		●
crashlytics2.l.google.com	●		●	●	●		●
firebaselogging-pa.googleapis.com	●		●	●	●		●
firebaseinstallations.googleapis.com	●		●	●	●		●
firebaselogging.googleapis.com	●		●	●	●		●
play.google.com	●		●	●	●		
settings.crashlytics.com	●		●		●		
github.com	●				●		●
firebaseremoteconfig.googleapis.com			●		●	●	
e.crashlytics.com	●		●		●		
api.crashlytics.com	●		●		●		
ns.adobe.com				●			
firebase-settings.crashlytics.com				●			●
update.crashlytics.com				●			●
certs.godaddy.com	●				●		
crashlyticsreports-pa.googleapis.com				●			●
www.apache.org		●					
plus.google.com		●					

Table 5. Common hostnames contacted by the apps analyzed.

A CORONAAPP (COLOMBIA)

A.1 App Metadata

App name	CoronApp
Package name	co.gov.ins.guardianes
Country	Colombia
Developer	Colombian INS
Certificate signer	GOOGLE INC.
Version	1.2.57
Min SDK	22
Target SDK	29
Hash (MD5)	b29f572d83725811e430e6d9cec34532
Implements GAEN	NO
Available on Google Play	YES
Privacy Policy	https://www.ins.gov.co/Normatividad/PoliticasInstitucionales/politica-de-tratamiento-de-informacion-coronapp-colombia.pdf

A.2 App Overview

This application was released by the Colombian government in collaboration with the national health service (Instituto Nacional de Salud, INS). It has been released for both Android and iOS.

According to the information available on Google Play, the application is meant to be used to control and monitor the spread of the virus, and to assist the national authorities in decision making. Specifically, it allows users to:

- Perform self-diagnosis and report symptoms (including those of their relatives) to the authorities. The application recommends that users report this information daily, including citizens without symptoms.
- Receive online medical assistance and information, as well as enabling mechanisms to assist citizens should lockdown be enforced (similar to the features observed in other applications internationally like Asistencia Covid-19 in Spain[105]).
- Access Covid-19 spread information in real-time.
- Monitor and enable citizens' mobility via QR codes.

According to the description of the app, the application implements contact-tracing capabilities using Bluetrace technology, a centralized implementation.[106] As a result, the application does not need to meet the privacy requirements set by Google to make use of Google's Exposure Notification.

Citizens need to provide a valid national ID number and a contact phone number to register and make full use of the app (excluding the access to public statistics and health information after an app upgrade released in late March 2020). As a result, users are not anonymous and they are required to expose sensitive information to the service to access most of the features.

Previous versions of the app have been exhaustively studied by **K+Lab**[107] using static and dynamic analysis techniques complementary to those used in this report. We refer the reader to the methodology description of this report for further details regarding our approach.

The application is implemented using Kotlin and Java code. It comprises of 40 activities, 9 services and 5 receivers. To the best of our knowledge, the application is not open source. Also, the application contains obfuscated code.

[105]https://blog.appcensus.io/2020/04/19/spanish-covid-19-apps/
[106]https://bluetrace.io/
[107]https://archive.org/details/informe-publico-tecnico-coron-app-v-170320-1/mode/2up

A.3 Permission Analysis

We identified the following permission requests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose.

Permission	Type	Protection Level
`android.permission.INTERNET`	AOSP	Normal
`android.permission.FOREGROUND_SERVICE`	AOSP	Normal
`android.permission.ACCESS_NETWORK_STATE`	AOSP	Normal
`android.permission.BLUETOOTH`	AOSP	Normal
`android.permission.BLUETOOTH_ADMIN`	AOSP	Normal
`android.permission.WAKE_LOCK`	AOSP	Normal
`android.permission.ACCESS_FINE_LOCATION`	AOSP	Dangerous
`android.permission.ACCESS_COARSE_LOCATION`	AOSP	Dangerous
`android.permission.CALL_PHONE`	AOSP	Dangerous
`com.google.android.c2dm.permission.RECEIVE`	Custom	Normal
`com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE`	Custom	Normal

Below, we discuss the use of three permissions of most concern from a privacy standpoint in this application, assessing whether they can cause any unnecessary privacy harm to users. We report in Section A.6 whether we have obtained evidence of this data being uploaded to a server.

A.3.1 ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION. We have identified different Application classes, Activities (*i.e.*, application components providing a user-interface) or Fragments requesting location data:

- `co.gov.ins.guardianes.manager.Application` Location is requested and stored in the class as soon as the application is launched.
- `co.gov.ins.guardianes.presentation.view.place.PlaceActivity` Location is requested to render users' location on a map.
- `co.gov.ins.guardianes.presentation.view.survey.SymptomActivity` Location is requested, along with user's ID.
- `co.gov.ins.guardianes.presentation.view.welcome.WelcomePageFragment` It checks whether users have enabled permission to access geolocation and if they have registered at the time of launching the application.

A.3.2 CALL_PHONE. This permission `android.permission.CALL_PHONE` is requested by the Fragment `co.gov.ins.guardianes.presentation.view.info.CallMinistryFragment` and by the Activity `co.gov.ins.guardianes.presentation.view.scheduled.ScheduleActivity`. In both cases, this permission is legitimately requested to contact the Covid-19 hotline set up by the Colombian health authorities.

A.3.3 Other observations. The presence of the Firebase-exposed permission `com.google.android.finsky.pe` on its `AndroidManifest.xml` file suggests that the app does integrate the third-party SDK Google Firebase, as also confirmed during the code inspection phase.

A.4 Third-party SDKs and Libraries

SDK	Type
Android Support v4	Development Support
ZXing ('Zebra Crossing')	Development Support
Glide	Development Support
Google Gson	Development Support
Fasterxml	Development Support
MaterialProgressBar	Development Support
Joda Time	Development Support
Material Dialog	Development Support
RxBinding	Development Support
OKHTTP	Development Support
ReactiveX	Development Support
Parceler	Development Support
Michael Rocks Libphonenumber	Development Support
Google Mobile Services	Tracking
Firebase	Tracking
Fabric	Tracking

The app integrates libraries that could cause privacy harm to end users, particularly Google Firebase (which integrates services like Google Analytics and Google Crashlytics, which is also integrated individually). A previous report by K+Lab [108]— using Exodus' privacy and traffic analysis— reported the presence of Facebook's SDKs in earlier versions, but we have not identified its presence in the tested version. We will report whether we identify any network flow to any SDK with tracking capabilities in the hostname analysis.

A.5 Hostnames and networking protocols

The application implements TLS-level security to protect users' data from network observers. According to a report by K+Lab,[109] earlier versions of the app uploaded sensitive data without using SSL/TLS standards like HTTPS.

Table 6 shows the hostnames identified by both static and dynamic analysis:

- The hostname `apicovid2.and.gov.co` is the base url to access Bluetrace online services.
- A set of auxiliary tools apparently used to assist users during a quarantine, e.g. learning platforms (`contenidos.colombiaprende.edu.co`), financial advice (`innpulsacolombia.com`), digital entrepreneurship (`apps.co`) or direct access to the Colombian police (`adenunciar.policia.gov.c`. All these are used in a class named *QuarantineHome*, and allow for direct access to these sites via regular web browsing.
- Two hostnames are used to obtain general information about the pandemic in Colombia. Concretely, the status map of the cases in Colombia (`arcgis.com`) and general government information (`coronaviruscolombia.gov.co`).
- The hostname `www.ins.gov.co` hosts the privacy policy.
- By dynamic analysis, we observe network activity from the app to Firebase servers, which confirms the use of this library for tracking purposes like installation monitoring.

[108] https://web.karisma.org.co/wp-content/uploads/2020/04/Informe-p%C3%BAblico-t%C3%A9cnico-CoronApp-v170320-1-1.pdf

[109] https://archive.org/details/informe-publico-tecnico-coron-app-v-170320-1/mode/2up

SRC	IP(s)	ASN(s)	CERT. INFO
		innpulsacolombia.com	
S	45.79.28.50	LINODE-AP Linode, LLC, US	CN=heroesfest.co
		apps.co	
S	34.202.254.4	AMAZON-AES, US	C=CH, L=Schaffhausen, O=Plesk, CN=Plesk/emailAddress=info@plesk.com
		coronaviruscolombia.gov.co	
S	190.145.219.66	Telmex Colombia S.A., CO	OU=Domain Control Validated, CN=coronaviruscolombia.gov.co
		contenidos.colombiaaprende.edu.co	
S	54.192.106.84 54.192.106.8 54.192.106.109 54.192.106.92	AMAZON-02, US	C=US, ST=Washington, L=Seattle, O=Amazon.com, Inc., CN=*.cloudfront.net
		apicovid2.and.gov.co	
S,D	3.229.237.212 52.54.29.154 3.225.120.50	AMAZON-AES, US	OU=Domain Control Validated, CN=*.and.gov.co
		www.ins.gov.co	
S	181.49.249.24	Telmex Colombia S.A., CO	businessCategory=Government Entity/serialNumber=899999403-4/jurisdictionC=CO, C=CO, ST=BOGOTA D.C., L=BOGOTA D.C./street=Avenida Calle 26 No. 51-20 - Zona 6 CAN, OU=OTIC, O=INSTITUTO NACIONAL DE SALUD, CN=ins.gov.co
		adenunciar.policia.gov.co	
S	190.255.40.104	COLOMBIA TELECOMUNICACIONES S.A. ESP, CO	C=CO, ST=CUNDINAMARCA, L=BOGOTA, O=POLICIA NACIONAL DE COLOMBIA, OU=Telematica, CN=*.policia.gov.co
		www.arcgis.com	

SRC	IP(s)	ASN(s)	CERT. INFO
S	34.199.206.244 18.234.22.28 18.234.22.251 52.23.2.231 34.233.149.104 18.234.22.27	AMAZON-AES, US	C=US, ST=California, L=Redlands, O=Environmental Systems Research Institute, Inc., CN=*.arcgis.com
crashlytics2.l.google.com			
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
settings.crashlytics.com			
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
app-measurement.com			
D	172.217.168.174	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
coronapp-97acb.firebaseio.com			
D	35.201.97.85	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=firebaseio.com
firebase.google.com			
D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
github.com			
D	140.82.121.4	GITHUB, US	C=US, ST=California, L=San Francisco, O=GitHub, Inc., CN=github.com
firebaselogging-pa.googleapis.com			
D	216.58.215.138	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
firebaseinstallations.googleapis.com			

SRC	IP(s)	ASN(s)	CERT. INFO
D	216.58.211.42	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaselogging.googleapis.com	
D	216.58.211.234	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		certs.godaddy.com	
D	173.201.201.4	AS-26496-GO-DADDY-COM-LLC, US	jurisdictionC=US/jurisdictionST=Arizona/businessC Organization/serial-Number=F20244620, C=US, ST=Arizona, L=Scottsdale, O=GoDaddy Inc., CN=mastercert.ext.pki.godaddy.com

Table 6. Hostnames potentially contacted found in the app COL_CoronApp_base.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis).

From our analyses, we do not observe personal data being sent to these sites due to the ethical limitations that prevent us from registering in the service. Also, all connections are encrypted using TLS/SSL.

A.6 Personal data dissemination

Given that the application requires user authentication, and the fact that the application is used to monitor the spread of the virus in real-time, our dynamic analysis of this software is limited for ethical and legal reasons. Therefore, we only test the application features that do not require user authentication or that can be identified via code inspection.

A.6.1 Android ID. The class `co.gov.ins.guardianes.manager.Application` requests access to the Android Advertising ID when the application is launched. We note a potential bad development practice in the request of this information, as the application developer prints the ID on the Android logcat, thus potentially exposing this information to other applications running on the same device. The code listing 3 shows the related code sample. Finally, the app developer is able to link the user's name, geolocation and ID number to Google's Android Advertising ID. The method to obtain this ID is also invoked by Firebase, Google Mobile Services, and Fabric.

```
public final String androidID() {
    String string = Settings.Secure.getString(getContentResolver(), "android\_id");
    Log.e("DEVICEID", string);
    Intrinsics.checkExpressionValueIsNotNull(string, "deviceId");
    return string;
}
```

Listing 3. Code sample accessing the Android Advertising ID (AAID), and printing it to Logcat. Class: co.gov.ins.guardianes.manager.Application, Lines 143 to 147

A.6.2 Geolocation. As previously discussed in the permission analysis, several application components request access to users' geolocation. While we could not find any evidence of geo-location data in the dynamic analysis performed (possibly due to the ethical considerations that prevent us from introducing data), code inspection suggests that this data is sent as users register on the service.

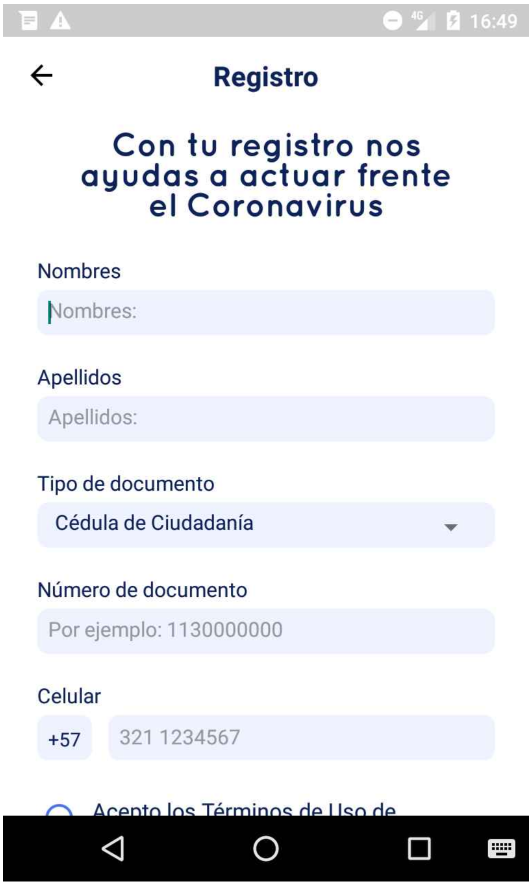
A.6.3 Personal and medical data. The application implements different forms to allow user registration. These forms require users to introduce sensitive personal data such as their first and family name, national ID number, and phone number which are uploaded to a server according to code inspection (we cannot confirm this due to the ethical limitations of our methodology). Figure 2 shows the registration form, corresponding to the *RegisterActivity*. This activity only contacts the server in `apicovid2.and.gov.co`, whose certificate belongs to `*.and.gov.co` and it is hosted by a US company (Amazon) where presumably the collected information is sent to.

A.7 Summary

This application implements features that could be considered privacy-intrusive, if compared to the most privacy-preserving contact-tracing applications developed in other world regions. The privacy policy of the app confirms the collection of personal data including geolocation, medical information, and personal data by the health authorities. It does not report, however, the presence of third-party tracking services like Google Firebase.

The use of contact-tracing features along with user geo-location and personal data collection goes against the privacy-by-design and data minimization requirements set by international organizations and scholars. In fact, a requirement set by GAEN to integrate this capability in contact-tracing apps is that applications must not attempt to geolocate users. The access to users' personal data and the use of safe passes could potentially enable mass surveillance and function creep.

Unfortunately, the lack of open-source code and the use of code obfuscation in parts of the app prevent us from producing an accurate assessment of the app behavior. These aspects, as well as the need to collect significant amounts of sensitive data, suggest that this app, despite its public interest, does not meet the highest international standards in terms of transparency.



16:49

Registro

Con tu registro nos ayudas a actuar frente el Coronavirus

Nombres

Nombres:

Apellidos

Apellidos:

Tipo de documento

Cédula de Ciudadanía

Número de documento

Por ejemplo: 1130000000

Celular

+57 321 1234567

Acepto los Términos de Uso de

Fig. 2. Registration form for the Colombian app

B CORONAVIRUS SUS (BRAZIL)

B.1 App Metadata

App name	Coronavírus SUS
Package name	br.gov.datasus.guardioes
Country	Brazil
Developer	DATASUS
Certificate signer	DATASUS
Version	2.1.6
Min SDK	21
Target SDK	29
Hash (MD5)	dc6882f3c42c81e3ec4c9bde4fdea1f3
Implements GAEN	YES
Available on Google Play	YES
Privacy Policy	https://validacovid.saude.gov.br/politica-privacidade

B.2 App Overview

Coronavirus SUS is an open-source contact-tracing application developed by the government of Brazil. Its code is publicly available on Github.[110] The application is implemented using web-based technologies (Apache Cordova and Capacitor JS) and is available for both iOS and Android.

The application uses the GAEN API. As a result, to comply with Google's privacy requirements, it does not collect any personal or geolocation data

The application offers the following features according to the description and implemented features:

- Contact-tracing using Google's Exposure Notification API.
- Provides health information about COVID-19 symptoms and recommendations for minimizing the risk of infection. It provides guidelines and information about how to proceed in case of presenting symptoms or if quarantine is necessary.
- A map showing nearby health centers.
- A news feed to offer citizens official news and relevant data related to the development of the COVID-19 pandemic in Brazil.

B.3 Permission Analysis

We identified the following permission requests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose.

Permission	Type	Protection Level
android.permission.RECEIVE_BOOT_COMPLETED	AOSP	Normal
android.permission.FOREGROUND_SERVICE	AOSP	Normal
android.permission.ACCESS_NETWORK_STATE	AOSP	Normal
android.permission.BLUETOOTH	AOSP	Normal
android.permission.INTERNET	AOSP	Normal
android.permission.WAKE_LOCK	AOSP	Normal
android.permission.VIBRATE	AOSP	Normal
android.permission.READ_APP_BADGE	Custom	Normal
com.anddoes.launcher.permission.UPDATE_COUNT	Custom	Normal
com.htc.launcher.permission.READ_SETTINGS	Custom	Normal
com.htc.launcher.permission.UPDATE_SHORTCUT	Custom	Normal
com.huawei.android.launcher.permission.CHANGE_BADGE	Custom	Normal

[110] https://github.com/spbgovbr/aplicativo-coronavirus-sus

com.huawei.android.launcher.permission.READ_SETTINGS	Custom	Normal
com.huawei.android.launcher.permission.WRITE_SETTINGS	Custom	Normal
com.majeur.launcher.permission.UPDATE_BADGE	Custom	Normal
com.oppo.launcher.permission.READ_SETTINGS	Custom	Normal
com.oppo.launcher.permission.WRITE_SETTINGS	Custom	Normal
com.sec.android.provider.badge.permission.READ	Custom	Normal
com.sec.android.provider.badge.permission.WRITE	Custom	Normal
com.sonyericsson.home.permission.BROADCAST_BADGE	Custom	Normal
com.sonymobile.home.permission.PROVIDER_INSERT_BADGE	Custom	Normal
me.everything.badger.permission.BADGE_COUNT_READ	Custom	Normal
me.everything.badger.permission.BADGE_COUNT_WRITE	Custom	Normal
com.google.android.c2dm.permission.RECEIVE	Custom	Normal

None of the permissions requested allows obtaining permission-protected sensitive data like geolocation, hardware IDs, or unique user identifiers. Overall, the app seems to comply with Google's Exposure Notification API privacy restrictions.

The observed custom permissions are related to proprietary usability APIs offered by devices running older Android versions, which is possibly the case of Brazil's mobile market. Their presence could be explained by developers' need to support a larger user-base.

B.4 Third-party SDKs and Libraries

This application integrates the following third-party SDKs:

SDK	Type
ZXing ('Zebra Crossing')	Development Support
Android Support v4	Development Support
Google Core Libraries for Java 6+	Development Support
Google Core Libraries (3rd Party)	Development Support
Google Protocol Buffers	Development Support
JavaX Annotation API	Development Support
Apache Common	Development Support
CapacitorJS	Development Support
Dagger	Development Support
LiteGlue	Development Support
Leolin	Development Support
Adobe Phonegap Push	Push Notifications / Development Support
Threteen	Development Support
Apache Cordova	Development Support
Google Mobile Services	Tracking / GAEN
Firebase	Tracking

Most of the third-party libraries and SDKs found in this app provide development support. The application is implemented using web technologies making use of Apache Cordova and Capacitor JS cross-platform development frameworks.

This application also integrates several libraries offered by Google, some of them offering tracking capabilities. Google Mobile Services (GMS) is legitimately required to integrate Google's Exposure Notification API. The presence of Firebase, however, is concerning due to its tracking capabilities. Its integration might be justified by the need to monitor application adoption (e.g., installation counts), but its presence opens unnecessary privacy risks as reported in previous COVID-19 app studies.[111] To meet the highest privacy-by-design and privacy-by-default requirements, it should be removed. It also integrates Adobe's Phonegap Push Notification service.

B.5 Hostnames and networking protocols

While there are some calls to HTTP URLs, code inspection suggests that the application uses TLS-based protocols like HTTPS. However, it does not seem to implement certificate pinning.

Table 8 shows information about the hostnames observed by both static and dynamic analysis:

- `exposure-export.saude.gov.br`. Is the base host to download and receive notifications into the phone.
- `exposure-notification.saude.gov.br`. Is the base host to upload exposure data to the server. Unlike the previous one, the certificate for this site is from Amazon.
- `unpkg.com`. is a content delivery network for software packages. Dynamic analysis suggests this is contacted from the MainActivity.
- `www.npmjs.org`. is the host of the Node Package Manager company, aimed at providing JavaScript packages. It is used in the MainActivity, while loading the 'cordova' third party library, in order to download additional JS packages.

Our dynamic analysis sandbox did not identify any active connection to Firebase and Adobe's PhoneGap servers. As a result, we cannot confirm whether these third-party services are actively used for tracking purposes or to use of some of their development support features. The stack trace analysis does not identify any call to Firebase's tracking capabilities.

SRC	IP(s)	ASN(s)	CERT. INFO
	exposure-notification.saude.gov.br		
S	54.192.106.49 54.192.106.82 54.192.106.116 54.192.106.7	AMAZON-02, US	C=US, ST=Washington, L=Seattle, O=Amazon.com, Inc., CN=*.cloudfront.net
	exposure-export.saude.gov.br		
S	52.67.50.120 18.230.83.60 18.230.30.82	AMAZON-02, US	CN=saude.gov.br
	unpkg.com		
D	104.16.124.175 104.16.122.175 104.16.123.175 104.16.126.175 104.16.125.175	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
	www.npmjs.org		

[111] https://www.nachdenkseiten.de/upload/pdf/200731-Leith+farrell-on-contact-tracing-app-privacy-july-2020.pdf

SRC	IP(s)	ASN(s)	CERT. INFO
D	104.16.21.35 104.16.25.35 104.16.22.35 104.16.26.35 104.16.27.35 104.16.23.35 104.16.19.35 104.16.20.35 104.16.17.35 104.16.24.35 104.16.16.35 104.16.18.35	CLOUDFLARENET, US	CN=ssl891738.cloudflaressl.com
		issues.apache.org	
D	168.119.33.54	HETZNER-AS, DE	OU=Domain Control Validated, OU=PositiveSSL Wildcard, CN=*.apache.org
		jsperf.com	
D	104.16.184.24 104.18.216.17	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
		www.apache.org	
D	40.79.78.1 95.216.24.32 95.216.26.30	MICROSOFT-CORP-MSN-AS-BLOCK, US HETZNER-AS, DE	CN=*.openoffice.org
		plus.google.com	
D	216.58.209.78	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com

Table 8. Hostnames potentially contacted found in the app Coronavirus_SUS_base.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis).

B.6 Personal data dissemination

We have not observed personal being sent to an external server, neither with static nor dynamic analysis.

B.6.1 Geolocation. The application includes Google Mobile Services (GMS), which might allow to get the fine and coarse location. Specifically, this is done in one method from the CapacitorJS library where various plugins are used, including one to get the location [112]:

[112] https://capacitorjs.com/docs/plugins/android

- com.getcapacitor.Bridge.registerAllPlugins()

However, the application does not have permission to access geolocation so this code will not provide any location fix.

B.6.2 Android ID. The application reads the Android ID with two methods listed below. However, these calls are related to the development frameworks Apache Cordova and CapacitorJS in order to populate a Java class storing device-specific information:[113]

- com.getcapacitor.plugin.Device.getUuid()
- org.apache.cordova.device.Device.getUuid()

B.7 Summary

The application seems to meet the highest privacy standards as it does not collect any sensitive personal data, such as persistent user identifiers and geolocation, and uses Google's Exposure Notification API. Our analyses reflect the claims in the privacy policy. Moreover, while our static analysis confirms that the app integrates third-party services with tracking capabilities, such as Google's Firebase, our dynamic analysis suggests that this popular SDK is used for non-tracking purposes.

The publication of the source code facilitates the analysis and impacts positively the transparency of the app.

To conclude, some may consider that the use of Cloudflare and Amazon cloud infrastructure – located in the U.S. – would raise concerns about data and digital sovereignty. Furthermore, the app does not implement dummy traffic to avoid identification and deanonymization of COVID-19 positive users that upload TEKs to the backend server. A description of this issue and how to avoid it can be found in DP3T's "Best Practices for Operation Security in Proximity Tracing" document.[114]

[113] https://capacitorjs.com/

[114] https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Best%20Practices%20for%20Operation%20Security%20in%20Proximity%20Tracing.pdf

C COVID ALERT SA (SOUTH AFRICA)

C.1 App Metadata

App name	COVID Alert SA
Package name	za.gov.health.covidconnect
Country	South Africa
Developer	NDOH
Certificate signer	NDOH
Version	1.4.0-gms
Min SDK	21
Target SDK	29
Hash (MD5)	5b67b99f97c2d73c8d5f7c08544ec0a5
Implements GAEN	YES
Available on Google Play	YES
Privacy Policy	https://sacoronavirus.co.za/covidalert/privacy-policy

C.2 App Overview

The COVID Alert SA contact-tracing application was released by the South African National Department of Health (NDOH). Its contact-tracing capabilities are implemented using Google's Exposure Notification API. Consequently, it must meet the privacy requirements imposed by Google (e.g., no collection of geo-location data).

The features offered by this app are a subset of those found on Coronavirus SUS (Brazil):

- Contact-tracing using Google's Exposure Notification API, allowing users to be notified of exposure.
- Contact details (24h COVID-19 hotline).

The application code is obfuscated. This is an accepted technique to protect the application code against software analysis (See listing below). However, this complicates the analysis of the app using code inspection techniques and goes against transparency criteria. At the time of writing, we did not identify any public repository containing the source code of the application, as exists for contact-tracing apps released by countries including Switzerland, Italy, and Spain. This practice allows the cybersecurity and technical community to inform developers about issues and even vulnerabilities present on their software.

```java
package c.a.a.a;

import android.app.Notification;
import android.os.Binder;
import android.os.IBinder;
import android.os.IInterface;
import android.os.Parcel;

public interface a extends IInterface {

    /* renamed from: c.a.a.a.a$a reason: collision with other inner class name */
    public static abstract class C0006a extends Binder implements a {

        /* renamed from: c.a.a.a.a$a$a reason: collision with other inner class name */
        public static class C0007a implements a {
            public IBinder a;
            public C0007a(IBinder iBinder) {
                this.a = iBinder;
            }

            public void N(String str, int i2, String str2, Notification notification) {
                Parcel obtain = Parcel.obtain();
                try {
                    obtain.writeInterfaceToken(
                                    "android.support.v4.app.INotificationSideChannel");
                    obtain.writeString(str);
                    obtain.writeInt(i2);
                    obtain.writeString(str2);
                    if (notification != null) {
                        obtain.writeInt(1);
                        notification.writeToParcel(obtain, 0);
                    } else {
                        obtain.writeInt(0);
                    }
                    this.a.transact(1, obtain, (Parcel) null, 1);
                } finally {
                    obtain.recycle();
                }
            }

            public IBinder asBinder() {
                return this.a;
            }
        }

        public static a e(IBinder iBinder) {
            if (iBinder == null) {
                return null;
            }
            IInterface queryLocalInterface = iBinder.queryLocalInterface(
                                    "android.support.v4.app.INotificationSideChannel");
            return (queryLocalInterface == null || !(queryLocalInterface instanceof a))
                ? new C0007a(iBinder) : (a) queryLocalInterface;
        }
    }

    void N(String str, int i2, String str2, Notification notification);
}
```

Listing 4. Code obfuscation.

C.3 Permission Analysis

We identified the following permission requests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose.

Permission	Type	Protection Level

android.permission.INTERNET	AOSP	Normal
android.permission.RECEIVE_BOOT_COMPLETED	AOSP	Normal
android.permission.FOREGROUND_SERVICE	AOSP	Normal
android.permission.ACCESS_NETWORK_STATE	AOSP	Normal
android.permission.BLUETOOTH	AOSP	Normal
android.permission.WAKE_LOCK	AOSP	Normal
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	Custom	Normal

None of the permissions requested by the app allows developers to access sensitive data or unique identifiers.

The presence of the custom permission BIND_GET_INSTALL_REFERRER_SERVICE in the app manifest file suggests the presence of Google's Firebase SDK. More details will be provided in the next section.

C.4 Third-party SDKs and Libraries

Despite the fact that the application code is obfuscated prior to its release, we could identify signals and fingerprints that suggest the presence of the following SDKs:

SDK	**Type**
Android Support v4	Development Support
Google GSON	Development Support
OKHTTP	Development Support
Babylon Health	Development Support
Google Mobile Services	Tracking / GAEN
Firebase	Tracking

The presence of the Firebase permission `BIND_GET_INSTALL_REFERRER_SERVICE` and different firebase URLs and packages in the code reveal that the app actively integrates Google's Firebase SDK.

C.5 Hostnames and networking protocols

Code analysis reveals potential hostnames and URLs associated with Google services like Firebase and Gstatic (Google's CDN).

Table 10 shows the hostnames contacted by the app:

- `api.whatsapp.com`. This is used to send messages using the WhatsApp Instant Messaging platform to a set of hard-coded numbers in the application.
  - +27 820468553 This number belongs to the National Department of Health (NDHO) of South Africa. Apparently, this phone number is used to receive COVID-19 test results [115]. From the code analysis, it is automatically sent with the following message: "ForgotPIN", which suggests that it is used to retrieve user's personal PIN.
  - +27 600123456 This number also belongs to the NDHO, and is used to contact COVIDConnect,the government's official COVID-19 support service.
- `www.gstatic.com`. Due to obfuscation, we cannot accurately identify the use of this domain. However, we did not capture any connection to this domain in our dynamic analysis.
- `files.ens.connect.sacoronavirus.co.za`. This is the download server from which information is gathered.

[115] https://sacoronavirus.co.za/2020/07/17/health-department-launches-covid-service-portal/

- `api.ens.connect.sacoronavirus.co.za`. This is the upload server. It is used in various classes to post information to the central server:

  `za.gov.health.covidconnect.forgotpin.ForgotPinActivity`.

  Although the code is obfuscated, we observed a suspicious URL being prepared, which suggests that the app might send the user's cell number, name and surname to this server. We discuss this observation further below.
- `static.xx.fbcdn.net`, `scontent.xx.fbcdn.net` and `mmx-ds.cdn.whatsapp.net` are three hosts contacted whenever the app interacts with Whatsapp's API. These hostnames belong to Facebook's Content Delivery Network (CDN) infrastructure.

Code inspection suggests that networking functions are implemented using secure network protocols like TLS. We observe invocations of Android's `TrustManager` as well as the integration of Babylon's Certificate Transparency library.[116] While these features make traffic-level analysis more difficult, they are a good security practice.

SRC	IP(s)	ASN(s)	CERT. INFO
		api.ens.connect.sacoronavirus.co.za	
S	63.33.9.219 52.49.126.24	AMAZON-02, US	CN=ens.connect.sacoronavirus.co.za
		api.whatsapp.com	
S,D	31.13.83.51	FACEBOOK, US	C=US, ST=California, L=Menlo Park, O=Facebook, Inc., CN=*.whatsapp.net
		www.gstatic.com	
S	172.217.17.3	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		files.ens.connect.sacoronavirus.co.za	
S	63.33.9.219 52.49.126.24	AMAZON-02, US	CN=ens.connect.sacoronavirus.co.za
		static.xx.fbcdn.net	
D	31.13.83.4	FACEBOOK, US	C=US, ST=California, L=Menlo Park, O=Facebook, Inc., CN=*.facebook.com
		mmx-ds.cdn.whatsapp.net	

[116] https://github.com/babylonhealth/certificate-transparency-android

SRC	IP(s)	ASN(s)	CERT. INFO
D	31.13.83.51	FACEBOOK, US	C=US, ST=California, L=Menlo Park, O=Facebook, Inc., CN=*.whatsapp.net
		scontent.xx.fbcdn.net	
D	31.13.83.4	FACEBOOK, US	C=US, ST=California, L=Menlo Park, O=Facebook, Inc., CN=*.facebook.com
		app-measurement.com	
D	172.217.168.174	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com

Table 10. Hostnames potentially contacted found in the app COVID_Alert_SA_base.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis).

C.6 Personal data dissemination

Unlike other applications, we could not identify any call to access non-protected data like the Android ID. However, its use cannot be completely discarded. The application does not access the user geolocation nor any hardware or user identifier.

However, our analysis of the hostnames (see Listing 5) yields a suspicious reference to a URL that potentially sends date of birth, cell number, name and surname to the backend servers. If this behavior is confirmed (e.g., by requesting that information to the app developers or by observing it using data captured from a real interaction), it would contradict the privacy claims reflected in Google Play's app description:

> *"Remember you always remain anonymous. You'll also get advice on what to do next to manage your health and to self-isolate."*

Due to obfuscation, we cannot confidently determine whether this data is actually being sent or not; and, if so, to which server. Dynamic analysis does not capture any connection to any URL containing this data. This may be due either to this item not being sent during the execution, or because it was sent encrypted and passed undetected.

```
Uri build = Uri.parse(getString(R.string.key_server_upload_uri)).
   buildUpon().appendPath(getString(R.string.forgot_pin)).build()
 [...]
String dVar = e.c.b.c.e.a().a(new SimpleDateFormat("yyyy-MM-dd").format((Date)
    Objects.requireNonNull(parse)), StandardCharsets.UTF_8).toString();
String str = (String) this.u.f.d();
String str2 = (String) this.u.d.d();
String str3 = (String) this.u.e.d();
 e eVar = new e(this, 1, String.format("%1s?dob=%2s&cellNo=%3s&name=%4s&surname=%4s",
     new Object[]{build, dVar, str, str2, str3}), null, new l.a.a.a.e.a(this), new
     l.a.a.a.e.b(this));
```

Listing 5. Code sample showing the potential URL being sent to the NDOH upload server, including cell number, name and surname Class: za.gov.health.covidconnect.forgotpin.ForgotPinActivity, Lines 217 and 224-228

The use of Whatsapp's API to notify app users the results of their COVID-19 tests raises privacy concerns, regardless of how convenient this may be. This approach might unnecessarily allow a third party with commercial interests to identify which users have been diagnosed as COVID-19 positives.

Our dynamic analysis did not capture any connection to Firebase for analytics purposes while it demonstrates that the library is actively being invoked. Therefore, as in the case of the Brazilian app, we do not have conclusive information to confirm or deny the use of this third-party SDK for tracking purposes.

C.7 Summary

This application follows Google's privacy requirements because it relies on Google's Exposure Notification API to implement contact-tracing capabilities. However, the app does not seem to implement dummy traffic to avoid identification and deanonymization of COVID-19 positive users that upload TEKs to the backend server. A description of this issue and how to avoid it can be found in DP3T's "Best Practices for Operation Security in Proximity Tracing" document.[117]

We have detected several behaviors that deserve further scrutiny:

- The presence of a suspicious obfuscated URL that might be used for uploading sensitive data.
- The use of Google's Firebase. It is unclear if it is used for tracking purposes. If so, this must be reflected in the applicaton's privacy policy.
- The use of Whatsapp's API to notify users of their COVID-19 test results. If this is confirmed, this behavior contradicts the points 3.1.2.c and 7.2.7 of the privacy policy:
  - 3.1.2.c: *Not even the NDoH, Discovery Limited or Telkom SOC Ltd are able to draw any conclusions concerning the identity of app users.*
  - 7.2.7: *People who test positive are not identified by the system to other users, to Apple or Google, or to the NDOH, Discovery or Telkom SOC.*

Making the code publicly available for public scrutiny is a good security and transparency practice, and could help clarify these concerns. Unless it is critical to protecting certain app features, code obfuscation is difficult to justify from a public interest perspective.

[117]https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Best%20Practices%20for%20Operation%20Security%20in%20Proximity%20Tracing.pdf

D MA3AN (LEBANON)

D.1 App Metadata

App name	Ma3an
Package name	com.tedmob.moph.tracer
Country	Lebanon
Developer	TEDMOB
Certificate signer	TEDMOB
Version	1.0.4
Min SDK	21
Target SDK	29
Hash (MD5)	3cb911a85b54cd496efe1fb06d8473be
Implements GAEN	NO
Available on Google Play	YES
Privacy Policy	https://mophtracer.tedmob.com/privacy-policy

D.2 App Description

Ma3an is Lebanon's official contact tracing and exposure notification app. It was developed jointly by the Ministry of Public Health (MoPH), the American University of Beirut, and the developer TedMob. According to its Google Play description, the app allow users *"to broadcast its presence anonymously to other mobile devices"*, and it also states that *"it does not collect any geolocation or GPS data"*. In order to register, the app requires the user to introduce a mobile phone number.

This application uses BlueTrace's technology, a centralized implementation of contact tracing.[118] According to the privacy policy, it works as follows:

(1) It generates a random code (not containing any personal or identifiable information) for your device and uses Bluetooth wireless communication to broadcast its presence anonymously to other nearby mobile devices. When your app detects another device with the Ma3an app, the devices exchange their respective random codes. The random numbers exchanged change every 15 minutes.
(2) The app does not store or collect location and operates on simple proximity detection.
(3) When a user tests positive for COVID-19, the MOPH will either send them a notification or provide them a code to upload their data on secure and encrypted MOPH servers, so to immediately notify other users who have been in close contact with the positive case. This happens without directly identifying positive users.

The application is implemented in Kotlin. It is composed of 14 activities, 17 services and 9 receivers. It only implements contact tracing and notification exposure using Bluetooth Low Energy. The source code is also available for public scrutiny.[119] Also, the app is not obfuscated. This shows efforts from the Lebanese Government to foster transparency. We note that this app requires users to register and grant access to geolocation data.

D.3 Permission Analysis

We identified the following permission requests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose.

Permission	**Type**	**Protection Level**
android.permission.INTERNET	AOSP	Normal

[118] https://bluetrace.io/
[119] https://github.com/Lebanese-Ministry-of-Public-Health/moph-ma3an-android

android.permission.RECEIVE_BOOT_COMPLETED	AOSP	Normal
android.permission.FOREGROUND_SERVICE	AOSP	Normal
android.permission.ACCESS_NETWORK_STATE	AOSP	Normal
android.permission.BLUETOOTH	AOSP	Normal
android.permission.BLUETOOTH_ADMIN	AOSP	Normal
android.permission.WAKE_LOCK	AOSP	Normal
android.permission.VIBRATE	AOSP	Normal
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	AOSP	Normal
android.permission.ACCESS_FINE_LOCATION	AOSP	Dangerous
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	Custom	Normal
android.permission.READ_APP_BADGE	Custom	Normal
com.anddoes.launcher.permission.UPDATE_COUNT	Custom	Normal
com.htc.launcher.permission.READ_SETTINGS	Custom	Normal
com.htc.launcher.permission.UPDATE_SHORTCUT	Custom	Normal
com.huawei.android.launcher.permission.CHANGE_BADGE	Custom	Normal
com.huawei.android.launcher.permission.READ_SETTINGS	Custom	Normal
com.huawei.android.launcher.permission.WRITE_SETTINGS	Custom	Normal
com.majeur.launcher.permission.UPDATE_BADGE	Custom	Normal
com.oppo.launcher.permission.READ_SETTINGS	Custom	Normal
com.oppo.launcher.permission.WRITE_SETTINGS	Custom	Normal
com.sec.android.provider.badge.permission.READ	Custom	Normal
com.sec.android.provider.badge.permission.WRITE	Custom	Normal
com.sonyericsson.home.permission.BROADCAST_BADGE	Custom	Normal
com.sonymobile.home.permission.PROVIDER_INSERT_BADGE	Custom	Normal
me.everything.badger.permission.BADGE_COUNT_READ	Custom	Normal
me.everything.badger.permission.BADGE_COUNT_WRITE	Custom	Normal
com.google.android.c2dm.permission.RECEIVE	Custom	Normal
com.tedmob.moph.tracer.permission.C2D_MESSAGE	Custom	Normal

The only dangerous permission requested by the app is geolocation, which we will discuss below. The presence of custom permissions related to usability aspects on low-end devices or devices running older Android versions—possibly the case in Lebanon's mobile market—could be explained by application developers' intent to support a larger user-base.

We also highlight the presence of different permissions for implementing push notifications, one declared by Tedmob (the application developer), and another based on Google's Cloud Messaging.

The presence of the Firebase-specific permission `BIND_GET_INSTALL_REFERRER_SERVICE` suggests that the app uses Firebase to collect installation analytics.

D.3.1 ACCESS_FINE_LOCATION. This permission is recommended by Google when using Bluetooth Services[120], although there are exceptions. We find this permission is also used by an external third-party library for tracking purposes. Listing 6 shows the code used in the OneSignal API to obtain the geolocation of the device. It contains various checks to see if the app has the permissions required, which in this case it does (concretely, the ACCESS_FINE_LOCATION). This code actually accesses the location, and is called whenever the device is registered to OneSignal servers together with other data (see below):

```
static void getLocation(Context context, boolean z, boolean z2, LocationHandler
    locationHandler) {
   [...]
```

[120]https://developer.android.com/guide/topics/connectivity/bluetooth

```
if (OneSignal.shareLocation) {
    String str = "android.permission.ACCESS_FINE_LOCATION";
    int checkSelfPermission = ContextCompat.checkSelfPermission(context, str);
    int i = -1;
    String str2 = "android.permission.ACCESS_COARSE_LOCATION";
    if (checkSelfPermission == -1) {
        i = ContextCompat.checkSelfPermission(context, str2);
        locationCoarse = true;
    }
    if (VERSION.SDK_INT < 23) {
        if (checkSelfPermission == 0 || i == 0) {
            sendAndClearPromptHandlers(z, PromptActionResult.PERMISSION_GRANTED);
            startGetLocation();
        } else {
            sendAndClearPromptHandlers(z,
                 PromptActionResult.LOCATION_PERMISSIONS_MISSING_MANIFEST);
            locationHandler.onComplete(null);
            return;
        }
    } else if (checkSelfPermission != 0) {
        try {
            List asList = Arrays.asList(context.getPackageManager().
                getPackageInfo(context.getPackageName(),
                     4096).requestedPermissions);
            PromptActionResult promptActionResult =
                 PromptActionResult.PERMISSION_DENIED;
            if (asList.contains(str)) {
                requestPermission = str;
            } else if (!asList.contains(str2)) {
                OneSignal.onesignalLog(LOG_LEVEL.INFO, "Location permissions not
                     added on AndroidManifest file");
                promptActionResult =
                     PromptActionResult.LOCATION_PERMISSIONS_MISSING_MANIFEST;
            } else if (i != 0) {
                requestPermission = str2;
            }
            if (requestPermission != null && z) {
                PermissionsActivity.startPrompt(z2);
            } else if (i == 0) {
                sendAndClearPromptHandlers(z,
                     PromptActionResult.PERMISSION_GRANTED);
                startGetLocation();
    [...]
```

Listing 6. Code to get location from onesignal. Class com.onesignal.LocationController, line 172

D.4 Third-party SDKs and Libraries

SDK	Type
Android Support v4	Development Support
Google Core Libraries (Guava)	Development Support
JavaX Annotation API	Development Support
Google Gson	Development Support
OKHTTP	Development Support
LibPhoneNumber	Development Support
Dagger	Development Support
AirBnB Lottie	Development Support
ReactiveX	Development Support
Google Mobile Services	Tracking
OneSignal	Tracking/Push
Firebase	Tracking
Fabric	Tracking

Upon inspection, the AndroidManifest file suggests that Firebase is used not only for tracking and analytics purposes (including the use of Crashlytics), but also for secure data storage and two-factor authentication (2FA).

It is unclear why the application makes use of multiple push notification services, including Google's Cloud Messaging, OneSignal and Huawei's push notifications. The use of OneSignal is concerning from a privacy standpoint given the sensitivity of the push notifications to be sent through a contact-tracing application, and the amount of personal and sensitive data that this SDK claims according to its privacy policy.[121] One can question the risks of exposing such sensitive user data to third-parties with commercial interests.

D.5 Hostnames and networking protocols

Both the static and dynamic analyses suggest that this application contacts a single hostname related to the application itself (*i.e.*, `mopthtracer.tedmob.com`). However, as shown in Table 12, it also contacts other hostnames related to third-party libraries such as Firebase and OneSignal or Google GMS.

- The hostname `mophtracer.tedmob.com` is the the main API entry point for the Ma3an Dashboard. According to the main site, this host is "Powered By `TEDMOB.com`", a company specialized in Mobile App development and services for Lebanon. We observe that this is hosted by a popular cloud provider from the U.S. – Amazon. This hostname is used twice in the code: one to either fetch (GET) an `upload_code` or to upload ("POST") a `player_id` (these two are integer variables). The other use of the hostname is to fetch the Privacy Policy from the server (in *PrivacyWebViewActivity*).

SRC	IP(s)	ASN(s)	CERT. INFO
		play.google.com	
S,D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		mophtracer.tedmob.com	

[121] https://onesignal.com/privacy_policy

SRC	IP(s)	ASN(s)	CERT. INFO
S,D	3.220.46.93	AMAZON-AES, US	OU=Domain Control Validated, CN=*.tedmob.com
		api.onesignal.com	
S,D	104.18.226.52 104.18.225.52	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
		firebasestorage.googleapis.com	
S	216.58.211.202	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		onesignal.com	
S,D	104.18.226.52 104.18.225.52	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
		crashlytics2.l.google.com	
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		settings.crashlytics.com	
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		moph-tracer.firebaseio.com	
D	35.201.97.85	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=firebaseio.com
		firebaseremoteconfig.googleapis.com	
D	172.217.168.170	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebase.google.com	

SRC	IP(s)	ASN(s)	CERT. INFO
D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		app-measurement.com	
D	172.217.168.174	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		github.com	
D	140.82.121.4	GITHUB, US	C=US, ST=California, L=San Francisco, O=GitHub, Inc., CN=github.com
		firebaselogging-pa.googleapis.com	
D	216.58.215.138	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaseinstallations.googleapis.com	
D	216.58.211.42	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaselogging.googleapis.com	
D	216.58.211.234	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		certs.godaddy.com	
D	173.201.201.4	AS-26496-GO-DADDY-COM-LLC, US	jurisdictionC=US/jurisdictionST=Arizona/businessC Organization/serial-Number=F20244620, C=US, ST=Arizona, L=Scottsdale, O=GoDaddy Inc., CN=mastercert.ext.pki.godaddy.com

Table 12. Hostnames potentially contacted found in the app Ma3an_base.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis).

D.6 Personal data dissemination

The application description on Google Play states that *“The app collects only the minimum amount of data, which is strictly necessary to support and improve the exposure notification system.”*. However, given the fact that the application requires users to register using their phone number, as well as grant access to geolocation, we conclude that the application does not follow the privacy-by-design and privacy-by-default principles, which are followed in other applications using Google's Exposure Notification API.

Listing 7 shows the information gathered from the devices, and sent to OneSignal servers when a user is registered. Among others, it contains the operating system and model of the device, the network carrier, the language, the time zone, and the location. This confirms that the data mentioned in OneSignal's privacy policy is actually being gathered from the devices running the app.

```
private static void registerUserTask() throws JSONException {
      String packageName = appContext.getPackageName();
      PackageManager packageManager = appContext.getPackageManager();
      JSONObject jSONObject = new JSONObject();
      jSONObject.put("app_id", getSavedAppId());
      if (getAdIdProvider() != null) {
          String identifier = getAdIdProvider().getIdentifier(appContext);
          if (identifier != null) {
              jSONObject.put("ad_id", identifier);
          }
      }
      jSONObject.put("device_os", VERSION.RELEASE);
      jSONObject.put("timezone", getTimeZoneOffset());
      jSONObject.put("language", OSUtils.getCorrectedLanguage());
      jSONObject.put(CommonUtils.SDK, VERSION);
      jSONObject.put("sdk_type", sdkType);
      jSONObject.put("android_package", packageName);
      jSONObject.put("device_model", Build.MODEL);
      try {
          jSONObject.put("game_version", packageManager.getPackageInfo(packageName,
               0).versionCode);
      } catch (NameNotFoundException unused) {
      }
      jSONObject.put("net_type", osUtils.getNetType());
      jSONObject.put("carrier", osUtils.getCarrierName());
      jSONObject.put("rooted", RootToolsInternalMethods.isRooted());
      OneSignalStateSynchronizer.updateDeviceInfo(jSONObject);
      JSONObject jSONObject2 = new JSONObject();
      jSONObject2.put(SettingsJsonConstants.APP_IDENTIFIER_KEY, lastRegistrationId);
      jSONObject2.put("subscribableStatus", subscribableStatus);
      jSONObject2.put("androidPermission",
           areNotificationsEnabledForSubscribedState());
      jSONObject2.put("device_type", osUtils.getDeviceType());
      OneSignalStateSynchronizer.updatePushState(jSONObject2);
      if (shareLocation) {
          LocationPoint locationPoint = lastLocationPoint;
          if (locationPoint != null) {
              OneSignalStateSynchronizer.updateLocation(locationPoint);
          }
      }
      OneSignalStateSynchronizer.readyToUpdate(true);
```

Listing 7. Code used by onesignal to prepare data for being sent to the servers. Class com.onesignal.OneSignal, line 1236

We observe that the library methods intended to get the user's email address are never called from the app, which suggests that either this information is collected by other means (e.g. reflection or native code), or it is not collected at all.

D.7 Summary

This application uses BlueTrace at its core for the contact tracing functionality. The app does not seem to access personal information itself, as declared in the privacy policy. The release of the source code, and the fact that this is not obfuscated in the code, is a good approach to increase transparency and, therefore, trust.

However, the inclusion of libraries such as Firebase and OneSignal is problematic from a privacy perspective. We observe that private user information is being sent to these two companies. According to the privacy policy, the permission *ACCESS_FINE_LOCATION* is required for Bluetooth to function properly in the app:

> "***Location information***: *Ma3an does not require your location to work. Android devices require the location permission to be granted in order for the app to access Bluetooth features. Your location data is not sent to our servers.*"

However, we observe that the location and other sensitive data is being sent to third-party stakeholders due to the inclusion of the *OneSignal* library. This behavior is not mentioned in the Privacy Policy.

E AAROGYA (INDIA)

E.1 App Metadata

App name	Aarogya Setu
Package name	nic.goi.aarogyasetu
Country	India
Developer	NITI AAYOG
Certificate signer	NITI AAYOG
Version	1.4.1
Min SDK	21
Target SDK	29
Hash (MD5)	fb1547abada4db53ceb20d04e652a3c1
Implements GAEN	NO
Available on Google Play	YES
Privacy Policy	https://web.swaraksha.gov.in/ncv19/privacy/,https://static.swaraksha.gov.in/tnc/

E.2 App Description

The Aarogya Setu app is a contact tracing app released by the Central Government of India. As most contact-tracing applications, it leverages Bluetooth Low Energy scans to identify neighboring devices.

According to the official documentation and the app description, it implements a large number of features:

- Contact-tracing using Bluetooth Low Energy technologies.
- Self-diagnose test based on ICMR guidelines and user-reported symptoms.
- Risk status of users (low, moderate or high-risk depending of users' exposure).
- Information, and official updates regarding best-practices related to COVID-19, including access to the national COVID-19 hotline.
- Providing telemetry and real-time statistics about the spread of the virus, including geo-location based statistics.
- ePass integration.
- Information of the list of officially accepted labs with COVID-19 testing facilities.
- An open API service to query information about the health status of employees or any other users (which requires user consent).

Compared to most contact tracing applications, the app collects an exceedingly large amount of personal data. The Indian government has released the source code of the application for public inspection and scrutiny [122], but not the code of the servers. The released application code is obfuscated with ProGuard.

It implements a centralized approach, in the sense that users' identity and their associated data (including geolocation) is stored in a central service.

In May 2020, SFLC – a digital rights organization – conducted a technical and security analysis on all available versions of the app. The last version analyzed was 1.1.1, whereas we report findings on a newer version (1.4.1) The following topics were looked at to determine the app's risk calculation: i) Application permissions; ii) Apkid analysis; iii) Manifest Analysis; iv) Code Analysis; v) Domain Malware Check; vi) URLs; vii) Firewall databases; viii) Emails; and viii) Trackers. Riddhi Shree, a cyber security researcher, has also reviewed (through a static code analysis) the android source code to understand how the app works and the types of data it collects. A technical analysis of the app using the code available on GitHub was also conducted to frame a set of recommendations in a recently released joint statement by three civil society organizations. One of the co-authors of

[122] https://github.com/nic-delhi/AarogyaSetu_Android

the letter had previously released a working paper that looked at the front-end features of the app, in addition to the privacy policy, terms of service, the permissions required by the app and data collected. The government released a technical FAQ[123] on the app's website, which addresses some of the "common issues flagged by code analysers or scanners." This may be useful to understand the claims raised in some of the other reports. Previous analyses have also been conducted to investigate the privacy risks of several apps released in India.[124]

The app does not rely on Google's Exposure Notification API and implements a custom centralized contact-tracing approach.

The application code is obfuscated. While this prevents us from conducting a reliable analysis, the source code of the application has been publicly released.[125] The app is composed of 15 activities, 13 services and 13 receivers. It is implemented using Kotlin and also Java Code, and uses various libraries (see below for a description of these).

E.3 Permission Analysis

We identified the following permission rquests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose, with the only difference being that `android.permission.RECEIVE_BOOT_COMPLETED` is not requested by this app.

Permission	Type	Protection Level
android.permission.INTERNET	AOSP	Normal
android.permission.FOREGROUND_SERVICE	AOSP	Normal
android.permission.ACCESS_NETWORK_STATE	AOSP	Normal
android.permission.BLUETOOTH	AOSP	Normal
android.permission.BLUETOOTH_ADMIN	AOSP	Normal
android.permission.WAKE_LOCK	AOSP	Normal
android.permission.ACCESS_BACKGROUND_LOCATION	AOSP	Dangerous
android.permission.ACCESS_FINE_LOCATION	AOSP	Dangerous
android.permission.ACCESS_COARSE_LOCATION	AOSP	Dangerous
android.permission.CAMERA	AOSP	Dangerous
com.google.android.c2dm.permission.RECEIVE	Custom	Normal
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	Custom	Normal

Below, we discuss the use of the following permissions and their capacity to cause privacy harm to users. We identify the purpose of these requests at the code level.

E.3.1 ACCESS_BACKGROUND_LOCATION, ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION. The use of location is pervasive across the app. Every discovered device is geolocated before storing it on the local database, as shown in listing 8. The ability to collect contact information with geolocation data can be harmful to privacy and enable function creep.

```
void storeDetectedUserDeviceInDB(BluetoothModel bluetoothModel) {
       if (bluetoothModel != null) {
```

[123]https://static.mygov.in/rest/s3fs-public/mygov_159056968451307401.pdf
[124]https://citizenmatters.in/tracking-quarantine-tracing-cases-sharing-info-can-these-govt-issued-apps-help-fight-covid-19-17151
[125]https://github.com/nic-delhi/AarogyaSetu_Android

```
                BluetoothData bluetoothData = new
                    BluetoothData(bluetoothModel.getAddress(), bluetoothModel.getRssi(),
                       bluetoothModel.getTxPower(), bluetoothModel.getTxPowerLevel());
                Location loc = CoronaApplication.getInstance().getAppLastLocation();
                if (loc != null) {
                    bluetoothData.setLatitude(loc.getLatitude());
                    bluetoothData.setLongitude(loc.getLongitude());
                }
                DBManager.insertNearbyDetectedDeviceInfo(bluetoothData);
            }

    }
```

Listing 8. Code sample showing how discovered devices are geo-tagged. Class: nic.goi.aarogyasetu.background.BluetoothScanningService

E.3.2 CAMERA. The camera is required since the app implements barcode scans, as shown in Listing 9

```
public void onResume() {
        super.onResume();
        String str = "binding";
        String str2 = "binding.barcodeScanner...._scanner.prompt_container";
        String str3 = "binding.barcodeScanner.custom_scanner";
        String str4 = "binding.barcodeScanner";
        ActivityCustomScannerBinding activityCustomScannerBinding;
        DecoratedBarcodeView decoratedBarcodeView;
        FrameLayout frameLayout;
        RelativeLayout relativeLayout;
        if ((ContextCompat.m1739a(this, "android.permission.CAMERA") == 0 ? 1 : null)
             != null) {
        [...]
```

Listing 9. Code sample showing the use of the CAMERA permission for barcodeScannerClass: nic.goi.aarogyasetu.qrcode.CustomScannerActivity

E.3.3 Other considerations. The use of the Google's Firebase specific permission `BIND_GET_INSTALL_REFERRER_SERVICE` confirms the presence of this third-party SDK and its use for collecting installation statistics and other usage telemetry.

E.4 Third-party SDKs and Libraries

SDK	Type
OKHTTP	Development Support
Zebra xing	Development Support
Android Support	Development Support
JourneyApps Barcodescanner	Development Support
Glide	Development Support
Google Mobile Services	Tracking / GAEN
Fabric	Tracking
Firebase	Tracking
Crashlytics	Tracking

In this app, the library GMS is used to access various services from Google, like *LocationRequest*, which is widely used throughout the code, or *GoogleSignIn*. The app uses three libraries from Google that are used for tracking,*i.e.*, Firebase, Crashlytics and Fabric.

E.5 Hostnames and networking protocols

SRC	IP(s)	ASN(s)	CERT. INFO
		api.swaraksha.gov.in	
S,D	3.7.209.156 35.154.255.228	AMAZON-02, US	CN=*.execute-api.ap-south-1.amazonaws.com
		fp.swaraksha.gov.in	
S,D	52.66.19.44 13.126.173.227 35.154.61.78 3.7.101.92 15.206.25.242 3.7.113.217 13.232.21.61 13.234.87.66	AMAZON-02, US	CN=*.cowin20.in
		web.swaraksha.gov.in	
S,D	15.207.70.82 13.126.56.147 35.154.83.99	AMAZON-02, US	CN=*.swaraksha.gov.in
		static.swaraksha.gov.in	
S,D	15.207.70.82 13.126.56.147 35.154.83.99	AMAZON-02, US	CN=*.swaraksha.gov.in
		d-05q3wc30r1.execute-api.ap-south-1.amazonaws.com	
D	13.235.170.239 52.66.149.194	AMAZON-02, US	CN=*.execute-api.ap-south-1.amazonaws.com
		static1.swaraksha.gov.in	
D	72.247.155.137 72.247.155.96	AKAMAI-ASN1, EU	C=US, ST=Massachusetts, L=Cambridge, O=Akamai Technologies, Inc., CN=a248.e.akamai.net
		ncv19-fe-alb-1549618494.ap-south-1.elb.amazonaws.com	

SRC	IP(s)	ASN(s)	CERT. INFO
D	15.207.70.82 13.126.56.147 35.154.83.99	AMAZON-02, US	CN=*.swaraksha.gov.in
		crashlytics2.l.google.com	
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		niti-app-alb-1092844402.ap-south-1.elb.amazonaws.com	
D	13.232.36.67 3.7.127.114 15.206.25.242 13.127.210.73 35.154.112.193 13.127.104.203 13.232.21.61 13.234.87.66	AMAZON-02, US	CN=*.cowin20.in
		a1838.dscb.akamai.net	
D	212.145.209.89 212.145.209.91 212.145.209.75 212.145.209.90 212.145.41.136 212.145.209.81 212.145.209.74 212.145.41.144 212.145.209.88	VODAFONE_ES, ES	C=US, ST=Massachusetts, L=Cambridge, O=Akamai Technologies, Inc., CN=a248.e.akamai.net
		settings.crashlytics.com	
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		app-measurement.com	
D	172.217.168.174	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		covid19-6c396.firebaseio.com	

SRC	IP(s)	ASN(s)	CERT. INFO
D	35.201.97.85	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=firebaseio.com
		firebase.google.com	
D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		firebaselogging-pa.googleapis.com	
D	216.58.215.138	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaseinstallations.googleapis.com	
D	216.58.211.42	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaselogging.googleapis.com	
D	216.58.211.234	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com

Table 14. Hostnames potentially contacted found in the app India_aarogya_setu.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis).

We observe that the app uses the okhttp library to handle TLS/SSL connections, and thus data transmission is secure. Table 14 shows the hostnames contacted by the app. These include various network flows to the tracking libraries from Google, which confirms that these are integrated and being used. The hostnames observed from the main app belong to the primary domain swaraksha.gov.in, and are hosted by external hosting providers from the U.S.

- `web.swaraksha.gov.in`. Used for various auxiliary purposes, e.g. to fetch the "Privacy Policy" document from the Home Activity. Also, it is an endpoint to delete an account (in which case the user_id is sent as part of the request)
- `fp.swaraksha.gov.in`. The usage is unclear. It is a hardcoded value used as default URL if no other is given to an obfuscated method in the class `p003e.p004a.p005a.p019m.NetworkClient`.
- `static.swaraksha.gov.in`. Used in to fetch the "Terms and Conditions" document from the Home Activity
- `api.swaraksha.gov.in`. Main endpoint used to fetch and send data. It requires an authentication token to interact with it. It is hosted in Amazon Web Services, the API KEY is hardcoded (ykixqE5H352HalBW4MNvI7HGJBXreLFx1APCkqEl), and it sends the server the SDK version, MANUFACTURER and MODEL when generating or validating One Time Passwords (OTPs) from the servers. The various endpoints (obtained from the class `e.a.a.m.d`) are:

  - "/api/v1/users/fcm/"
  - "/api/v1/openapi/userpref/remove/"
  - "/api/v1/openapi/msme/verify/"
  - "/api/v1/openapi/msme/initiate/"
  - "/api/v1/users/register/"
  - "/api/v3/users/data/"
  - "/api/v1/account/delete/"
  - "/api/v1/openapi/userpref/"
  - "validateOTP"
  - "/api/v1/openapi/approval/"
  - "generateOTP"
  - "/api/v1/openapi/granter/remove/"
- `static1.swaraksha.gov.in` This hostname was observed during dynamic analysis. It was accessed from the classes *HomeActivity*, *SettingsActivity* and *DeleteAccountActivity*. Unlike other subdomains under the main domain `'swaraksha.gov.in'`, this one is hosted by a different hosting provider *Akamai*.

The hosts (ending in amazonaws.com and akamai.net) are only observed during DNS resolution. These are requests needed to contact the hosting providers of the previously mentioned hosts.

E.6 Personal data dissemination

Whenever a new user is registered in the web, the app sends the following information: OS version (SDK), manufacturer, model, device type, and version of the app. Also, among the Bluetooth data sent (from the contacted peers), it also sends the Bluetooth unique address (MAC) and location info. Using BT MAC address as an identifier is a bad practice, since this address is revealed automatically to other devices as part of the BT discovery process, and this information can be used to uniquely identify users. The internal table `nearby_devices_info_table` stores the following information:*userid*, *bluetooth_mac_address*, *distance*, *latitude*, *longitude*, *timestamp*, *tx_power*, *tx_power_level*, and *is_uploaded*. This information is uploaded to the server (class UploadDataUtil) using the API endpoint `/api/v3/users/data/`. However, due to obfuscation we cannot assess whether this information is always sent, or if users can opt out from sending this amount of information. Manual inspection of the source code suggests that users cannot opt out.

E.7 Summary

Our analysis of this app suggests a significant number of privacy concerning behaviors that can enable function creep.

According to the privacy policy, at the time of registration the app stores the following information in an encrypted form on the server: i) Name; ii) Gender; iii) Age; iv) Profession; v) Countries visited in the last 30 days; and vi) Willingness to volunteer in times of need. It also collects location data at the point of registration, which is also stored in the server. Additionally, at the time of filling in the self-assessment form, the app captures the user's location data. This, along with the results and unique device ID, is stored in the server. It has also been specified that the app

> *"continuously collects your location data and stores securely on your mobile device, a record of all the places you have been at 15 minute intervals."*

This information is only uploaded to the server in specific cases, as detailed by the privacy policy.

It has also been claimed that the app follows a *"Privacy-by-Design principle"* and that *"the app never discloses any personal information related to its users."* According to official sources, the following privacy measures have been taken: i) the information provided at the time of app registration is anonymised and subsequent transactions are made with a specific Device Identification Number (DiD); ii) all the contact tracing and location information collected is stored locally on the user's mobile device. The information is only uploaded to the server if the user tests positive.

In terms of security measures, the privacy policy states that *"data is encrypted from in transit as well as at rest."* It has also been claimed that *"All interactions between two devices that have the Aarogya Setu app installed, and between the device and the Aarogya Setu server are done using DiD only. No personal information is used for any communication or transaction."* Additionally, officials have claimed that all data is encrypted and anonymized: The re-identification of anonymized information only takes place only if medical intervention is necessary.

However, the unique identifier used to track contacts is the physical Bluetooth MAC address. This is a bad privacy practice since this address might be revealed to anyone as part of a Bluetooth advertising process. Best privacy practices suggest the use of ephemeral identifiers, *i.e.*, those that change frequently.[126]

In general, we observe that most of the data is transmitted to hosts that are subdomains of *swaraksha.gov.in*. These are hosted by two well-known hosting provider companies, *Amazon* and *Akamai*. We do not know what commercial relationships exist between the Indian Government and these entities to access the servers. The data in transit is secured since the protocol SSL is always used. Also, access to the API requires authentication, and it uses One Time Passwords.

In May 2020, a security analyst who goes by the pseudonym Elliot Anderson reported privacy and security issues in the app. (He explains them in detail in a blogpost). The Aarogya Setu team acknowledged his efforts but denied any such vulnerabilities. Obfuscation of the app hinders analysis. While the source code of the application is available, we have observed some differences between the decompiled code (obfuscated) from the app, and the source code released. For example, the class (deobfuscated with *Jadx*) `p003e.p004a.p005a.p023q.UploadDataUtil` differs from the source code released publicly. This might be either due to errors in the decompilation process, or because the application has changed (the source code was released in May 2020, while the version analyzed is dated July 2020).

It is unclear to us how appropriate the consent form shown to users is – particularly if it reflects the data collected and its purposes – because we were unable to check it.

The integration of third-party SDKs with tracking and monitoring capabilities (Firebase) is also concerning from a privacy perspective.

[126]https://github.com/DP-3T/documents/raw/master/DP3T%20White%20Paper.pdf

F COVA (INDIA)

F.1 App Metadata

App name	COVA
Package name	in.gov.punjab.cova
Country	India
Developer	GOVT OF PUNJAB
Certificate signer	GOVT OF PUNJAB
Version	1.3.23
Min SDK	19
Target SDK	29
Hash (MD5)	8bdff3a52b84b9a9ee44130bd88ce70e
Implements GAEN	NO
Available on Google Play	YES
Privacy Policy	https://msewa.punjab.gov.in/m-sewa/privacyPolicy.html

F.2 App Description

The COVA application was developed by the Government of Punjab to assist citizens with preventive care information and broadcast official announcements in the Punjab region. Notifications from the government are sent via push notifications. According to the GooglePlay description, the app provides the following capabilities:

(1) Real time dashboard for Punjab, India and global stats
(2) To check for symptoms of Corona and have a quick self-screening
(3) Corona Awareness
(4) Travel Instructions
(5) Prevention Products
(6) Corona Hospitals, Punjab
(7) FAQ
(8) Call Support

We note that the app contains other capabilities, like access to the Mission Fateh program[127], which motivates citizens to take appropriate measures to control the spread of the virus, and provides them with official certificates when a set of milestones are reached.

According to an existing report by Rohini Lakshane, an individual researcher, the app requires users to register an account with an Indian mobile number, and does not implement contact tracing functionality.[128] However, an advisory from the Punjab government from August 20, 2020, includes contact tracing as one of the app's capabilities .[129] Another report by Srikanth Lakshmanan using the MobFS security framework states that the app has certain security weaknesses.[130] Finally, a report from SFLC indicates the permissions requested (*i.e.*, Location, Photos/Media/Files, Storage, Camera and Others) and also the personal information accessed (Personal demographic, Location, Device information, Usage details and User submissions).[131]

The official documentation from Google Play does not indicate whether the app includes contact tracing functionality or not. But a report from the Punjab government says that the app does

[127]http://lgpunjab.gov.in/cms/mission-fateh.php
[128]https://citizenmatters.in/tracking-quarantine-tracing-cases-sharing-info-can-these-govt-issued-apps-help-fight-covid-19-17151
[129]https://dgrpg.punjab.gov.in/wp-content/uploads/2020/08/20.08.2020-COVID-RESPONSE-PUNJAB.pdf
[130]https://docs.google.com/spreadsheets/u/1/d/e/2PACX-1vS9a4eNZ-8w8QVs9qkqqOwiGP6B9TeIsf8d3FG1igRr03Iaby-HTbOYJdbdHwEbrZ5E4Vd0i4qKo6m8/pubhtml
[131]https://sflc.in/our-analysis-indian-covid19-apps

'contact tracing of positive cases' [132]. A notification from the Directorate of Information and Public Relations from Punjab confirms that the app *'gives notifications if a covid positive or suspected case is around the app user'.* [133]

The application is developed using Kotlin and Java code. It is a complex application, composed of 49 Activities, 10 Services and 4 Receivers. This reflects the multiple capabilities described in the official sources. The app is obfuscated, and it implements other anti-analysis techniques, like String obfuscation, debugger and emulator checks, and the use of NOP instructions in methods. To the best of our knowledge, the source code has not been released publicly. This hinders static analysis and limits public scrutiny.

F.3 Permission Analysis

We identified the following permission requests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose. The permission requests are almost identical to those requested by Aarogya Setu app.

Permission	**Type**	**Protection Level**
android.permission.INTERNET	AOSP	Normal
android.permission.FOREGROUND_SERVICE	AOSP	Normal
android.permission.ACCESS_NETWORK_STATE	AOSP	Normal
android.permission.BLUETOOTH	AOSP	Normal
android.permission.BLUETOOTH_ADMIN	AOSP	Normal
android.permission.WAKE_LOCK	AOSP	Normal
android.permission.ACCESS_BACKGROUND_LOCATION	AOSP	Dangerous
android.permission.ACCESS_FINE_LOCATION	AOSP	Dangerous
android.permission.ACCESS_COARSE_LOCATION	AOSP	Dangerous
android.permission.CAMERA	AOSP	Dangerous
com.google.android.c2dm.permission.RECEIVE	Custom	Normal
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	Custom	Normal

Below, we discuss the use of the following permissions according to their capacity to cause privacy harm to users. We identify the purpose of these requests at the code level.

F.3.1 ACCESS_BACKGROUND_LOCATION, ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION. The location is a key component of this application. It is used for example to maintain the perimeter distance when a user is in quarantine (*i.e.*, to create a so-called *Geofence* area). It is also used to check the distance maintained with other users of the application. If this distance is lower than the threshold (by default set to 10 meters) with any suspected or infected user, the app notifies the user. The main wrapper to obtain location is implemented in the class `a.a.a.a.g.f.b` (see Listing **??**. This class is widely in the code.

```
public b(Context context) {
  if (context != null) {
      this.f = context;
      this.b = new a(this);
      Object systemService = this.f.getSystemService("location");
      if (systemService != null) {
          this.a = (LocationManager) systemService;
```

[132]https://dgrpg.punjab.gov.in/wp-content/uploads/2020/08/20.08.2020-COVID-RESPONSE-PUNJAB.pdf
[133]http://www.diprpunjab.gov.in/?q=content/cova-app-our-tool-fight-against-corona-arvindpal-singh-sandhu

```
            LocationManager locationManager = this.a;
            if (locationManager != null) {
                LocationListener locationListener = this.b;
                if (locationListener != null) {
                    locationManager.requestLocationUpdates("gps", 3000, 0.0f,
                         locationListener);
                    return;
    }
```

Listing 10. Code sample accessing the LocationManager service. Class: a.a.a.a.g.f.b, Lines 24 to 36

F.3.2 CAMERA. The use of the camera is justified in this case. It is used in various classes for taking photographs, *i.e.*, to report self-assessment. It is also used to scan QR codes. We have not found evidence of this permission being used for different means than those reported in the official documentation.

F.4 Third-party SDKs and Libraries

SDK	Type
Android-SpinKit	Development Support
Dhaval2404 ImagePicker	Development Support
Glide	Development Support
CodeScanner	Development Support
SiliCompressor	Development Support
Nabinbhandari Permission Manager	Development Support
Soffritti YoutubePlayer	Development Support
RootBeer	Development Support
FilePicker	Development Support
uCrop	Development Support
Google Mobile Services	Tracking

The application includes various development libraries obtained from the GitHub software repository. Concretely, most libraries are intended to improve the management of media content (*i.e.*, Android-SpinKit, Dhaval2404 ImagePicker, Silicompressor, FilePicker and uCrop). Then, the CodeScanner library is used to assist in the scanning of QR codes, in the `PassDashboardFrame` class (*i.e.*, to scan health-related ePassports). The Android-Permissions library is used to assist managing permissions granted dynamically. YoutubePlayer is used to embed videos into one of the Activities. Finally, RootBeer is used to check whether the device is rooted or not in the class `a.a.a.a.a.i`, and if so, notify users with the following message: *"Your device is not compatible to use this app. Contact support to know more"*.

The GMS library is used to provide Location services, Maps, Google Sign-in and also Firebase's Google Mobile Ads.[134]

F.5 Hostnames and networking protocols

Table 17 shows the hosts contacted by this app. Next, we describe the use of these within the app:

- `covidhelp.punjab.gov.in`. This web provides information to travelers moving to and from Punjab. Concretely, in the app it is used to show the Terms and Conditions for hotel booking due to mobility restrictions [135]. It is also used to register *Labour*-related information. The last functionality has two main security concerns.

[134]https://firebase.google.com/docs/reference/android/com/google/android/gms/ads/package-summary
[135]http://covidhelp.punjab.gov.in/HotelBookingTermandConditions.aspx

  - Lack of confidentiality: The communications are not secure (*i.e.*, data is not encrypted). We observe that the URL sends the geolocation data and telephone number of the user.
  - Lack of authentication. The authentication key (`AUTH_KEY`) is hard-coded in the app [136]

  The URL returns a form that must be filled by the user, which is sent in cleartext over HTTP, and includes sensitive information such as the address, bank details or family photograph (see Figure 4)
- `cova.punjab.gov.in`. This API endpoint mostly contains access to the MissionFateth program, which encourages users to follow government guidelines and provides an official certificate for those completing a set of milestones. The connections to the MissionFateth are not secure, since HTTPS is not used, and includes sending users' images and registration details. This is confirmed by dynamic analysis. (see Figure 3). Below we describe the personal data sent through this API.
- `covaprod.punjab.gov.in`. This is the main API endpoint, used for a variety of purposes (see Table 16), including to get data from nearby users, register a traveler, get hospital information, etc.
- `msewa.punjab.gov.in`. Additional API endpoint for the command `populate-entities`.
- `esewa.punjab.gov.in`. This is the Digitital Punjab portal. It is used to register a new appointment automatically with one of the Departments in the portal.
- `hrms.punjab.gov.in`. Main portal of Human Resources Management System. Used to apply for leave, and also for general information about services.
- `pblabour.gov.in`. Main portal for the Department of Labour. It is used to apply for a "*Registration of Shop and Commercial Establishments under The Punjab Shops and Commercial Establishments Act, 1958 in Punjab.*"
- `dronamaps.com`. This is an end-to-end platform to collect, process and visualize drone data. It is used to create a local map of the region with stats such as the number of people in quarantine. It provides the exact location of people who are home-quarantining. In the app, it is used to show such information when requested by the user.
- `epasscovid19.pais.net.in`. At the time of writing, this site cannot be reached so we cannot assess what it is used for.
- `esanjeevaniopd.in`. This is the National TeleConsultation service for health information offered by the Indian Government. It is included as a direct link from the app.
- `daily-needs.uen.io`. This portal is used to order food from various groceries.
- `play.google.com`. Google Play Store, used to check for new versions of the app
- There are two hosts related to Azure, Microsoft's cloud services (those ending in `cloudapp.azure.com`). We observe these only by dynamic analysis. These hosts are contacted from various Frames in the app (*e.g.*, *SelfInspectionFrame*, *HomeLocationFrame* or *PasswordChangeFrame*), but only the DNS is resolved and we cannot monitor further interaction with these services.
- Finally, there are two hosts contacted during dynamic analysis that are related to the tracking library *Firebase* and also the ads Network *DoubleClick*, both from Google.

SRC	IP(s)	ASN(s)	CERT. INFO
		www.google.com	

[136] U6JOqCMyOLsK84GwfBLR6pbQ7kevmZJ7Xd15OFwDkRmAddWjE9dm08s5EY2VuvdUAROEwn1ii1paaYSc

SRC	IP(s)	ASN(s)	CERT. INFO
S,D	172.217.17.4	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=www.google.com
daily-needs.uen.io			
S	15.206.6.139	AMAZON-02, US	CN=addo.uengage.in
pblabour.gov.in			
S	164.100.60.214	NICNET-VSNL-BOARDER-AP National Informatics Centre, IN	businessCategory=Government Entity/jurisdictionC=IN/serialNumber=Government Entity, C=IN, ST=Punjab, L=Chandigarh, O=Department of Labour Punjab, OU=IT, CN=pblabour.gov.in
www.google.com			
S,D	172.217.17.4	GOOGLE, US	Not applicable
cova.punjab.gov.in			
S,D	52.172.8.28	MICROSOFT-CORP-MSN-AS-BLOCK, US	C=IN, ST=Punjab, O=Governance Reforms Department, Punjab, OU=Department of Governace Reforms, CN=*.punjab.gov.in
esewa.punjab.gov.in			
S	104.211.222.199	MICROSOFT-CORP-MSN-AS-BLOCK, US	CN=esewa.punjab.gov.in
covaprod.punjab.gov.in			
S,D	20.44.43.14	MICROSOFT-CORP-MSN-AS-BLOCK, US	C=IN, ST=Punjab, O=Governance Reforms Department, Punjab, OU=Department of Governace Reforms, CN=*.punjab.gov.in
dronamaps.com			
S	142.93.108.123 167.99.129.42	DIGITALOCEAN-ASN, US	C=US, ST=ca, L=San Francisco, O=Netlify, Inc, CN=*.netlify.com
hrms.punjab.gov.in			

SRC	IP(s)	ASN(s)	CERT. INFO
S	103.118.160.28	DGRPB-AS-IN Department of Governance Reforms, IN	C=IN, ST=Punjab, O=Governance Reforms Department, Punjab, OU=Department of Governace Reforms, CN=*.punjab.gov.in
csi.gstatic.com			
S	216.58.199.131	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
epasscovid19.pais.net.in			
S	NOT FOUND		Not applicable
play.google.com			
S,D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
esanjeevaniopd.in			
S	3.7.29.155 13.126.1.10	AMAZON-02, US	CN=esanjeevaniopd.in
imasdk.googleapis.com			
S,D	216.58.212.106	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
msewa.punjab.gov.in			
S	13.71.89.83	MICROSOFT-CORP-MSN-AS-BLOCK, US	CN=msewa.punjab.gov.in
cova.punjab.gov.in			
S,D	52.172.8.28	MICROSOFT-CORP-MSN-AS-BLOCK, US	Not applicable
www.youtube.com			

SRC	IP(s)	ASN(s)	CERT. INFO
S,D	216.58.211.46 216.58.209.78 216.58.215.142 172.217.168.174 172.217.17.14 216.58.201.174	GOOGLE, US	Not applicable
		covidhelp.punjab.gov.in	
S	20.44.32.249	MICROSOFT-CORP-MSN-AS-BLOCK, US	Not applicable
		covaprodlb.southindia.cloudapp.azure.com	
D	20.44.43.14	MICROSOFT-CORP-MSN-AS-BLOCK, US	C=IN, ST=Punjab, O=Governance Reforms Department, Punjab, OU=Department of Governace Reforms, CN=*.punjab.gov.in
		covaprodadmin.southindia.cloudapp.azure.com	
D	52.172.8.28	MICROSOFT-CORP-MSN-AS-BLOCK, US	C=IN, ST=Punjab, O=Governance Reforms Department, Punjab, OU=Department of Governace Reforms, CN=*.punjab.gov.in
		crashlytics2.l.google.com	
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		firebase-settings.crashlytics.com	
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		app-measurement.com	
D	172.217.168.174	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		cova-275dc.firebaseio.com	

SRC	IP(s)	ASN(s)	CERT. INFO
D	35.201.97.85	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=firebaseio.com
		update.crashlytics.com	
D	54.221.212.159 54.225.187.80	AMAZON-AES, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=update.crashlytics.com
		firebase.google.com	
D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		googleads.g.doubleclick.net	
D	216.58.211.34	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.g.doubleclick.net
		crashlyticsreports-pa.googleapis.com	
D	216.58.201.131	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		firebaselogging-pa.googleapis.com	
D	216.58.215.138	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaseinstallations.googleapis.com	
D	216.58.211.42	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaselogging.googleapis.com	
D	216.58.211.234	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		clients4.google.com	

SRC	IP(s)	ASN(s)	CERT. INFO
D	172.217.17.14	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com

Table 17. Hostnames potentially contacted found in the app India_cova_app.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis). Highlighted in red are those using HTTP

F.6 Personal data dissemination

Using static analysis, we can spot various personal data being sent. However, due to the complexity of the code and the fact that it is obfuscated (both the code and some Strings), this analysis might be incomplete.

In general, the app provides various forms for the user to fill in, *e.g.*, to ask for a permission to travel or to access the MissionFateth program. The MissionFateth Activity uses a non-secure communication channel, and thus information is sent without encryption. Concretely, the app asks the user for the following information during registration: Mobile Number, Name, Address and Photograph. Also, it automatically gathers the userID from the app .

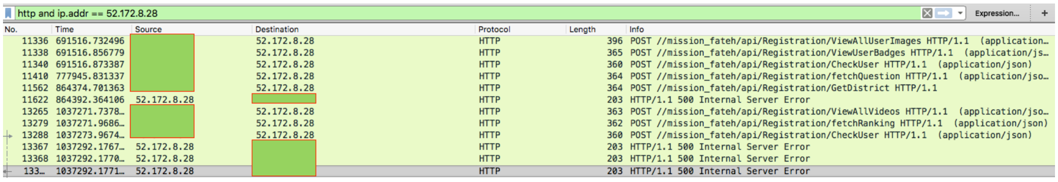
Registering with government services requires the phone number and full name, and in return it provides a `userToken` field, identifying the device. This identifier is used as an authentication mechanism whenever there is an interaction with the main API (covaprod.punjab.gov.in). Also, often the data sent to the server includes geolocation data, *e.g.*, when the user initiates the Bluetooth service which scans for nearby contacts, or when a self-quarantine inspection is carried out, to verify that the user is keeping within the *geofence* distance established.

F.7 Summary

COVA is a multi-purpose application intended to help in the fight against the Covid pandemic in the Punjab region of India. It also has contact-tracing capabilities. However, unlike other applications analysed in this document, Cova app notifies users when they are close to other users who are either suspected or confirmed as having COVID-19. It uses a centralized approach, since the information is sent to and requested from a central server owned by the government. Thus, it constantly tracks users by means of the Bluetooth communications, and also connections to the server to post and fetch information needed in the protocol.

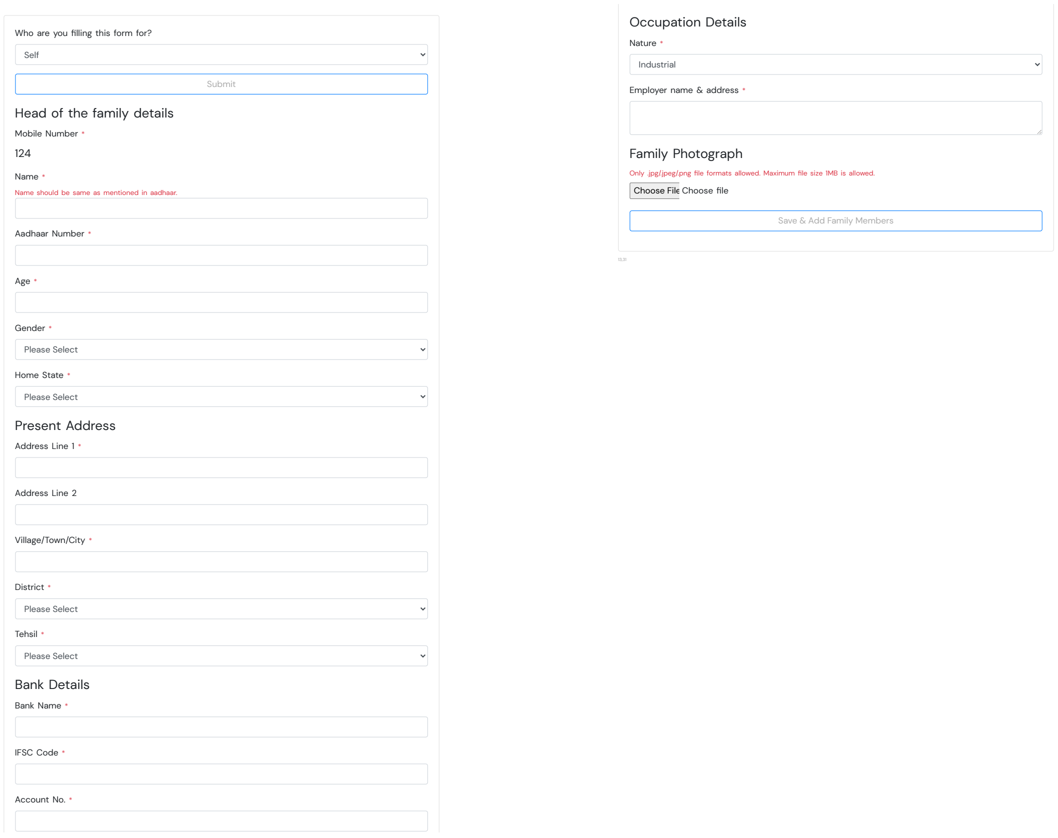
According to the privacy policy, the following information *'may be collected: personal, demographic, location, device and other similar information'*. The collection of such information is confirmed in our analyses, although we were not able to assess faithfully whether other information might be being collected due to the app being obfuscated. The privacy policy also states that *'data are encrypted using SSL as permissible under the law'*. However, sensitive information from the users willing to participate in the MissionFateh program is sent without protection.

The app includes various open-source libraries from GitHub for development and support. While using open-source code is a popular practice, if might allow potential privacy abuse or security bugs due to bad coding practices by external developers. One of such libraries is used to dynamically grant permissions. Our analysis shows that no further permissions apart from those requested in the Manifest are required. Still, the use of these might be obfuscated in the code, and our dynamic analysis might not have reached the point where such new permissions are requested. Finally, the application includes third-party libraries used for tracking users, *i.e.*, Firebase; therefore, the personal data being collected is also transmitted to third-party entities.



```
http and ip.addr == 52.172.8.28                                                              Expression…  +
No.    Time             Source        Destination   Protocol   Length  Info
11336  691516.732496    [redacted]    52.172.8.28   HTTP       396  POST //mission_fateh/api/Registration/ViewAllUserImages HTTP/1.1  (application…
11338  691516.856779    [redacted]    52.172.8.28   HTTP       365  POST //mission_fateh/api/Registration/ViewUserBadges HTTP/1.1  (application/js…
11340  691516.873387    [redacted]    52.172.8.28   HTTP       360  POST //mission_fateh/api/Registration/CheckUser HTTP/1.1  (application/json)
11410  777945.831337    [redacted]    52.172.8.28   HTTP       364  POST //mission_fateh/api/Registration/fetchQuestion HTTP/1.1  (application/jso…
11562  864374.701363    [redacted]    52.172.8.28   HTTP       364  POST //mission_fateh/api/Registration/GetDistrict HTTP/1.1
11622  864392.364106    52.172.8.28   [redacted]    HTTP       203  HTTP/1.1 500 Internal Server Error
13265  1037271.7378…    [redacted]    52.172.8.28   HTTP       363  POST //mission_fateh/api/Registration/ViewAllVideos HTTP/1.1  (application/jso…
13279  1037271.9686…    [redacted]    52.172.8.28   HTTP       362  POST //mission_fateh/api/Registration/fetchRanking HTTP/1.1  (application/json)
13288  1037273.9674…    [redacted]    52.172.8.28   HTTP       360  POST //mission_fateh/api/Registration/CheckUser HTTP/1.1  (application/json)
13367  1037292.1767…    52.172.8.28   [redacted]    HTTP       203  HTTP/1.1 500 Internal Server Error
13368  1037292.1770…    52.172.8.28   [redacted]    HTTP       203  HTTP/1.1 500 Internal Server Error
133…   1037292.1771…    52.172.8.28   [redacted]    HTTP       203  HTTP/1.1 500 Internal Server Error
```

Fig. 3. Traffic capture from dynamic analysis showing unencrypted communications with the MissionFateh API endpoint on IP 52.172.8.28



Who are you filling this form for?

Self

Head of the family details

Mobile Number *

124

Name *

Name should be same as mentioned in aadhaar.

Aadhaar Number *

Age *

Gender *

Please Select

Home State *

Please Select

Present Address

Address Line 1 *

Address Line 2

Village/Town/City *

District *

Please Select

Tehsil *

Please Select

Bank Details

Bank Name *

IFSC Code *

Account No. *

Occupation Details

Nature *

Industrial

Employer name & address *

Family Photograph

Only .jpg/.jpeg/.png file formats allowed. Maximum file size 1MB is allowed.

Fig. 4. Form requested in the app, all sent insecurely (obtained using the dummy URL: http://covidhelp.punjab.gov.in/web-views/RegisterLabour.aspx?authkey=U6JOqCMyOLsK84GwfBLR6pbQ7kevmZJ7Xd15OFwDkRmAddWjE9dm08s5EY2VuvdUAROEwn1ii1paaYSc&mobno=1234&geocoords=12,34)

Overall, this application does not follow Privacy-by-Design principles. It collects personal data and sends it both to government-owned and third-party stakeholders. Also data transmission is insecure in some aspects. There are serious deficiencies regarding transparency. While the government is aware of the potential security weaknesses reported by previous reports, and have published due responses, the code is still not public, and the app is highly obfuscated.

populate-entities	get-app-version	register-citizen	resend-otp
update-ios-device	validate-otp	verify-and-resend-otp	Genrate_Token
Get-token-list	GetValue	Post-feedback	bluetooth-insert-data-for-all-users
check-train-booking	covid-help-fetch-registration	covid-help-update-board-status	covid-help-update-board-status-for-train
fetch-icmr-result	get-All-Notification	get-COVA-Symptoms-list	get-Passes-Type
get-Services-Type	get-Sold-Item	get-Token-Purpose-list	get-bluetooth-notification-msg
get-common-master-data	get-district-passes	get-globally-stats	get-googlemap_flag
get-hospital-list	get-hot-spot-areas	get-hq-data-district-wise	get-patient-data
get-patient-stats	get-punjab-district-wise-data	get-track-flag-data	get-train-master-data
get-traveller-register-details	get-user-detail	insert-COVA-Symptoms	insert-COVA_MASS_Gathering
insert-Distance	insert-blo-data	insert-bluetooth-app-data	insert-district-passes
insert-emp-attendance	insert-hq-user-info	insert-plazma-donor	insert-train-booking-data
patient-hotspot-hq-count-data-nearby	report-inter-state-traveller	sync-app-data	update-advertising-data
update-citizen-home-location	update-moment-status	verify-sms-icmr_result	view-kft-pass-by-id-mobile
view-pass-by-id	view-pass-by-mobile	get-emp-att-data	get-emp-att-weekly-data
update-emp-password	validate-emp-hrms	authenticate-employee	fetch-private-hospital-list
search-hq-data	travellerreg		

Table 16. Commands available for the API endpoint covaprod.punjav.gov.in

G MASK (IRAN)

G.1 App Metadata

App name	Mask
Package name	ir.covidapp.android
Country	Iran
Developer	COVIDAPP
Certificate signer	COVIDAPP
Version	2.0.2.0
Min SDK	16
Target SDK	28
Hash (MD5)	9f1abcf624e4986ff618bebbc97651f0
Implements GAEN	NO
Available on Google Play	NO
Privacy Policy	https://www.mask.ir/privacy.html

Important remark. We note that the application analyzed in this report has been downloaded in a Western European country using official sources.[137] We cannot discard the possibility that the application downloaded and studied in this report might be different from the app downloaded directly in Iran. To confirm the validity of the studied app (and whether it has been distributed in Iran), it is necessary to compare the MD5 hash.

G.2 App Description

Mask is an official app that provides Iranian citizens with different information, notification, and movement-tracing services related to the COVID-19 pandemic. The app was developed for the Ministry of Health (MoH) by a team of volunteers –mostly technical experts and specialists in medical sciences– from Sharif, Amirkabir, and Shahid Beheshti Universities. It is available on the mask.ir website[138] and the Café Bazaar app store.[139] It was also published on Google Play but removed due to sanctions against the country.[140]

The app implements the following key features:

- An infection-risk map showing the outbreak of the pandemic. According to the official website, the map is based on various sources:
  - The latest official data obtained from the Ministry of Health about patients, which includes their approximate place of residence.
  - Location data obtained from users and mobile network operators (MNOs).
  - User-provided data through an on-board contact and infection-risk notification service.

  Users can view the map without registering with the app
- Official COVID-19 information. The nature of this information is unknown and is not reported. Users can navigate through these pages without registering with the app.
- A self-assessment and proximity-based risk notification service. To use these services, users must register by providing their phone number and authorize access to their device's location. The exposure notification service is centralized and does not implement any anonymization technique whatsoever. A backend server receives the phone number of positive-diagnosed users from the MoH. Using the location data of confirmed cases, it informs registered app users who have been in close contact with Covid-19 positive users in the last two weeks. The

[137]https://mask.ir/application.html
[138]https://mask.ir
[139]https://cafebazaar.ir/app/ir.covidapp.android?l=en
[140]https://www.tasnimnews.com/en/news/2020/05/16/2266827/google-play-removes-iranian-app-used-against-coronavirus

website also states that exposure notification employs information provided voluntarily by users.

The application is developed using Kotlin and Java code. It contains a substantial amount of components: 28 activities, 12 services, 5 providers, and 18 receivers. This might be explained by the variety of features it offers to users.

The app shows a number of anti-analysis features: the class names and methods have been obfuscated, it uses reflection quite heavily, uses NOP instructions in multiple methods, and checks for the presence of debuggers. These efforts make app analysis harder, and can prevent the accurate identification of potentially harmful behaviors.

We did not identify any public repository containing the source code of the application.

G.3 Permission Analysis

We identified the following permission requests in the AndroidManifest.xml file. Please see Section 4.3 for a detailed description of the permissions and their purpose.

Permission	Type	Protection Level
android.permission.ACCESS_NETWORK_STATE	AOSP	Normal
android.permission.CAMERA	AOSP	Dangerous
android.permission.INTERNET	AOSP	Normal
android.permission.RECEIVE_BOOT_COMPLETED	AOSP	Normal
android.permission.VIBRATE	AOSP	Normal
android.permission.WAKE_LOCK	AOSP	Normal
com.google.android.c2dm.permission.RECEIVE	Custom	Normal
com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	Custom	Normal

Below, we discuss the use of specific permissions and their capacity to cause privacy harm to users.

G.3.1 CAMERA. The use of the Camera in this app is justified since it is needed for scanning QR codes, which is a fundamental feature of the application. The camera is used by some of the SDKs reported later on this section (ZXing Android Embedded and the barcode scanner). We have not found evidence of privacy-harmful behaviors arising from the use of this permission. However, we cannot reliably discard it due to the limitations of our analysis because of the use of code obfuscation and other anti-analysis techniques.

G.3.2 On the Use of LOCATION. Two location permissions (ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION permissions) are checked in the *GeoUtils* and *NetworkInfoHelper* classes (names known after de-obfuscation by jadx), both from the *pushe* package. In addition, the code has multiple artifacts suggesting it accesses the device location. For example, the package `c.a.a.p` imports `android.location.Location` and retrieves the location. The `LocationManager` component is also used by the `l.b.k.t` class. However, the version of the applications analyzed does not request those permissions in the manifest.

The presence of location-related functionality might be due to legacy code left from previous versions, where the location was actually requested in the manifest. Also, this is expected since the app description explicitly acknowledges that it does access the device location. The privacy policy[141] also includes a similar statement confirming the use of location data (*"it only has access to your phone's location ..."*).

[141] https://www.mask.ir/privacy.html

G.3.3 Access to External Storage. As in the case of location described above, the app uses functionality that requires granting of two dangerous permissions: READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE.

These are used at runtime, but are not declared in the Manifest. In addition, we have not found any evidence in the code of these being requested by the user. This requires further analysis.

G.4 Third-party SDKs and Libraries

The application integrates the following third-party SDKs:

SDK	Type
Android Support v4	Development Support
AndroidX	Development Support
Kotlin / KotlinX	Development Support
Calligraphy	Development Support
Google Material	Development Support
OKHTTP 3.0	Development Support
Google Gson	Development Support
PinView	Development Support
EventBus	Development Support
Moshi	Development Support
PersianMaterialDateTimePicker	Development Support
ZXing Android Embedded	Development Support
MPAndroidChart	Development Support
Neshan	Development Support
Pushe.co	Tracking
Sentry	Tracking
Google Mobile Services	Tracking / GAEN
Firebase	Tracking

The use of the development SDKs listed above is explained by the features the app exhibits and the implementation choices of the developers. The presence of SDKs for analytics and tracking purposes (Firebase, GMS, Sentry and Pushe.co) is concerning from a privacy perspective. Their use might be motivated by a need to monitor app adoption (e.g., installation counts), the use of app features, or to report errors back to developers. However, the presence of such extensive tracking capabilities implies an unnecessary privacy risk for users of an app of this nature. It therefore does not meet the expected privacy-by-default / privacy-by-design requirements.

G.5 Hostnames and networking protocols

Table 19 shows the hostnames in this app found by static analysis . We note that most of them are unique to this app due to the use of the Iranian library *pushe.co*. Also, most of the URLs related to this library are only observed through static analysis, *i.e.*, they were not visited during the dynamic execution of the app.

- `barkat.shop`. This hostname is observed statically in the *pushe* library as part of the app notifications. It is used to download an APK, which at the time of this writing was not available in the URL. Apparently, the downloaded app (`shop.barkat.android`) is used to provide cooking recipes and ingredients. The communication is not encrypted.
- `khandevaneh.arsh.co`. Also observed statically in the *pushe* library as part of the notifications of the app. It downloads the app `co.arsh.khandevaneh`, which has been removed from the Café Bazaar app store at the request of the developer, but is still available from the main page.

- `files.softicons.com`. Used by the *pushe* library to download an icon for image-based notifications. Communications are not encrypted.
- The *pushe* library contains an internal list of public IP APIs, that are used to get the IP of the device:
  - `api.ipify.org`
  - `ip-alt.pushe.co`
  - `ip.pushe.co`
  - `ip-seeip.org`
  - `ipapi.co`
- `static.pushe.co`. Used by the *pushe* library. This host is contacted to retrieve a list of 'blacklisted' system applications (`system_apps_black_list.json`), which at the time of writing contains applications installed by default in various devices, like *com.facebook.system* and also OEM [142] related apps (*e.g.*, *com.samsung.android.messaging*). We ignore what this list is used for in the library.
- `api.pushe.co`. Used in *pushe* library to get a PNG file used as a visual (image) notification.
- `notificationsounds.com` Used by the *pushe* library to get a file (in *MP3* format) to use as a sound notification.
- `api.covidapp.ir`. Main API endpoint for the app. It is used for the following processes:
  - `/event` Used to notify different events, like persons contacted.
  - `/user`. Used to manage registration and user accounts. Concretely, it has the following sub-paths:
    - `/token/revoke`
    - `/token/refresh`
    - `/profile`
    - `/register`
    - `/activate/code`
    - `/people`
    - `/health_guard/people/verify`
    - `/people/:id`
    - `/people/:id/share`
    - `/push_token`
    - `/generate_qr_code`
    - `/health_state_by_qr_code`
  - `/question` Used to answer questions related to health status, like reporting fever.
  - `/map`
- `capi.covidapp.ir`. API endpoint used to send (POST) information to the server
- `cdn.api.covidapp.ir`. API endpoint used to retrieve the following metadata information related to the app: *about*, *config* and *version*
- `cdn2.api.covidapp.ir`. API endpoint used to retrieve current information about the pandemic, concretely the current infection map (*maps.json*) and statistics about infections (*infected.json*)
- `api.neshan.ir`. Neshan is a Persian map service. By dynamic analysis, we observe this host is contacted from the *HomePage* activity
- `vts1.neshanmap.ir`. This host is also contacted from the *HomePage* activity,
- `www.arvancloud.com`. This is the main page for the hosting provider of various of the hosts analyzed previously. It is observed during dynamic analysis.

[142]Original Equipment Manufacturer

SRC	IP(s)	ASN(s)	CERT. INFO
khandevaneh.arsh.co			
S	213.233.161.229	SHARIF-EDU-NET, IR	CN=grapefruit.arsh.co
capi.covidapp.ir			
S,D	185.143.233.5 185.143.234.5	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=covidapp.ir
cdn.covidapp.ir			
S,D	185.143.233.5 185.143.234.5	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=*.arvancloud.com
ipapi.co			
S	104.26.8.44 172.67.69.226 104.26.9.44	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
ip.pushe.co			
S	104.31.67.200 104.31.66.200 172.67.152.70	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
api.pushe.co			
S	104.31.67.200 104.31.66.200 172.67.152.70	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
ip-alt.pushe.co			
S	104.31.67.200 104.31.66.200 172.67.152.70	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
files.softicons.com			
S	208.76.175.41	CIFNET, US	Not applicable
cdn2.covidapp.ir			
S,D	185.143.233.5 185.143.234.5	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=*.arvancloud.com
corona-269507.firebaseio.com			

SRC	IP(s)	ASN(s)	CERT. INFO
S,D	35.201.97.85	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=firebaseio.com
api.covidapp.ir			
S,D	185.143.233.5 185.143.234.5	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=covidapp.ir
api.ipify.org			
S	23.21.42.25 54.243.161.145 184.73.247.141 54.225.66.103 54.235.83.248 23.21.126.66 54.243.164.148 54.204.14.42	AMAZON-AES, US	OU=Domain Control Validated, OU=PositiveSSL Wildcard, CN=*.ipify.org
static.pushe.co			
S	104.31.67.200 104.31.66.200 172.67.152.70	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
notificationsounds.com			
S	104.27.130.102 172.67.201.22 104.27.131.102	CLOUDFLARENET, US	C=US, ST=CA, L=San Francisco, O=Cloudflare, Inc., CN=sni.cloudflaressl.com
barkat.shop			
S	172.67.143.95 104.27.169.83 104.27.168.83	CLOUDFLARENET, US	Not applicable
ip.seeip.org			
S	23.128.64.141	JOESDATACENTER, US	CN=ip.seeip.org
crashlytics2.l.google.com			
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
firebase-settings.crashlytics.com			

SRC	IP(s)	ASN(s)	CERT. INFO
D	216.58.201.163	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		vts1.neshanmap.ir	
D	185.143.233.212 185.143.234.212	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=*.arvancloud.com
		api.neshan.org	
D	185.143.233.212 185.143.234.212	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=*.arvancloud.com
		app-measurement.com	
D	172.217.168.174	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.google.com
		update.crashlytics.com	
D	54.221.212.159 54.225.187.80	AMAZON-AES, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=update.crashlytics.com
		github.com	
D	140.82.121.4	GITHUB, US	C=US, ST=California, L=San Francisco, O=GitHub, Inc., CN=github.com
		crashlyticsreports-pa.googleapis.com	
D	216.58.201.131	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=google.com
		firebaselogging-pa.googleapis.com	
D	216.58.215.138	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		firebaseinstallations.googleapis.com	
D	216.58.211.42	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com

SRC	IP(s)	ASN(s)	CERT. INFO
		firebaselogging.googleapis.com	
D	216.58.211.234	GOOGLE, US	C=US, ST=California, L=Mountain View, O=Google LLC, CN=*.googleapis.com
		www.arvancloud.com	
D	185.143.233.5 185.143.234.5	ABRARVAN-AS AbrArvan CDN and IaaS, IR	CN=*.arvancloud.com

Table 19. Hostnames potentially contacted found in the app mask_v2.0.2.0.apk. SRC column is the source of the information (S=Static analysis, D=Dynamic analysis). Highlighted in red are those using HTTP

G.6 Personal data dissemination

The privacy policy lists other data items collected. The first category refers to data manually provided by the user, which includes:

- Phone number. This is requested by a form and sent to the server during registration of the user (using the `/user/register` API endpoint), together with the risk group.
- Places the user registers to visit. This is done by the *pushe.co* library in a GeofenceManager class.
- Appointments recorded via a QR code or Bluetooth. We did not find where this information is collected.
- Health-related information entered on a daily basis. This is done by means of a form which requests health status and text.
- Answers provided to the questions made by the app. These are requested by means of a question form and sent to the servers using the `/question` API endpoint.
- If the user registers any family members, their phone number or national ID number are collected through a form and sent to the server using the `/user/person` API endpoint

The second category refers to data automatically collected by the app, which encompasses:

- Location. This is collected by pushe.co library.
- Appointments via Bluetooth. However, the necessary permission is not requested in the app, and Bluetooth is indeed not used as a service in the source code.
- IP and User Agent in communications to the backend server. These data are sent automatically because of the HTTPS protocol.

The app accesses the following pieces of sensitive data:

- Android ID (AID)
- Service Provider Name
- Device model
- Android version
- Contacts
- Location

G.7 Summary

Here we provide an assessment of this app based on the findings reported above.

The app does not comply with the Privacy-by-Design and Privacy-by-Default principles. The collection of personal data, including user geolocation and health data items of a potentially highly sensitive nature, does not respect the data minimization principle. The exposure notification service is centralized and does not implement any anonymization technique. The access to users' personal data in such a centralized design could potentially enable mass surveillance and function creep.

In addition, the app integrates multiple third-party SDKs with analytics and tracking capabilities. This unnecessarily increases privacy risks and potential for abuse. Overall, these design choices are not aligned with the requirements and best privacy practices set by international organizations and scholars.

The privacy policy[143] acknowledges the collection of personal data but does not provide sufficient provisions in several key aspects. For example, it is vague about the data retention period:[144]

> *Mask* ***will try to delete your information*** *as soon as it becomes useless in terms of fighting Corona.*

However, it also explicitly states that it could be used to fight other "infectious diseases":

> *Privacy is one of our red lines, and the developers of Mask* ***promise*** *to use user data solely to control the spread of corona* ***and other infectious diseases****.*

It acknowledges that some data will be shared with the health authorities:

> *It should be noted that only the data that you enter in the "Daily Health Information Entry" section* ***will be shared with the Ministry of Health*** *so that the Ministry of Health can provide better services to you by attaching them to your electronic health record.*

Remarkably, data collected by the app can also be used for law enforcement purposes:

> *Also, if a legal order is issued by the judicial authorities, Mask will be obliged to present your information to the court.*

The privacy policy does not describe the presence of other parties in the code that can be considered data processors, such as the SDKs with tracking and monitoring capabilities described above.

The supervisory and regulatory mechanisms in place to monitor the appropriate use of the collected data are unclear:

> *The same information that you provide remains completely confidential, and committees composed of representatives of some of the most reputable specialized scientific, civic and independent institutions in the country monitor the confidentiality of your data.*

The app is also deficient in other transparency aspects. The source code is not public and the app has been packaged with anti-analysis techniques. This impedes an accurate analysis by making it difficult to do a reliable assessment of the app's true functionality and capabilities.

[143]https://www.mask.ir/privacy.html

[144]Translation and emphasis in all quotes from the privacy policy are ours.