Contact Tracing App Privacy: What Data Is Shared By Europe's GAEN Contact Tracing Apps

Douglas J. Leith, Stephen Farrell
School of Computer Science & Statistics,
Trinity College Dublin, Ireland
18th July 2020

Abstract*—We describe the data transmitted to backend servers by the contact tracing apps now deployed in Germany, Italy, Switzerland, Austria, Denmark, Spain, Poland, Latvia and Ireland with a view to evaluating user privacy. These apps consist of two separate components: a "client" app managed by the national public health authority and the Google/Apple Exposure Notification (GAEN) service, that on Android devices is managed by Google and is part of Google Play Services. We find that the health authority client apps are generally well behaved from a privacy point of view, although the privacy of the Irish, Polish, Danish and Latvian apps could be improved. In marked contrast, we find that the Google Play Services component of these apps is extremely troubling from a privacy viewpoint. In one "privacy conscious" configuration, Google Play Services still contacts Google servers roughly every 20 minutes, potentially allowing fine-grained location tracking via IP address. In addition, Google Play services also shares the phone IMEI, hardware serial number, SIM serial number, handset phone number, the WiFi MAC address and user email address with Google, together with fine-grained data on the apps running on the phone. This data collection is enabled simply by enabling Google Play Services, even when all other Google services and settings are disabled. It therefore appears to be unavoidable for users of GAEN-based contact tracing apps on Android. This level of intrusiveness seems incompatible with a recommendation for population-wide usage. We note the health authority client app component of these contact tracing apps has generally received considerable public scrutiny and typically has a Data Protection Impact Assessment, whereas no such public documents exist for the GAEN component of these apps. Extending public governance to the full contact tracing ecosystem, not just of the health authority client app component, therefore seems to be urgently needed if public confidence is to be maintained.

I. Introduction

Countries across Europe are currently rolling out mobile apps to facilitate Covid-19 contact tracing. This is motivated by the hope that more efficient and scalable contact tracing might allow the lockdown measures in place in many countries to be relaxed more quickly [1] and that these systems can help "hedge" against the risk of a second wave of the pandemic [2]. In early April 2020, Apple and Google formed a partnership to develop contact event detection based on Bluetooth LE [3]. Following public launch of the Google/Apple Exposure Notification (GAEN) API on 20 May 2020 [4], GAEN implementations are now installed on many people's phones and this API is starting to be widely used by national health authority contact tracing apps.

The pressures imposed by the ongoing pandemic and the timeline above mean that all this development has taken place under severe time-pressure with many decisions having to be made speedily in the face of significant uncertainty. We therefore approach this measurement exercise with a view to highlighting potential improvements, and not from a perspective of attempting to allocate blame. We believe that all of the actors involved are attempting to do their best in a challenging situation. That said, given that many governments are encouraging entire populations to use these apps it is necessary that the detail of their operation be visible to enable properly informed choices by users and potential users of these apps.

We measure the actual data transmitted to backend servers by the GAEN-based contact tracing apps now deployed in Germany [5], Italy [6], Switzerland [7], Austria [8], Denmark [9], Poland [10], Latvia [11] and Ireland [12] with a view to evaluating user privacy. We also take measurements with the RadarCOVID [13] app currently being trialled in Spain. To the best of our knowledge this is the first examination of the privacy of the overall GAEN service as deployed to date. In this report we focus on an independent assessment of data shared by the Android implementations of contact tracing apps based on the GAEN API. We look forward to similar work being done with the Apple app implementations and of Apple's underlying handling of privacy whilst GAEN apps are in use.

It is important to note that all of these apps consist of two separate components, under the control of different parties. Firstly, a "client" app managed by the national public health authority. This client app provides the user interface. It also interacts with nationally managed back-end servers to (i) allow the upload of handset keys (TEKs in GAEN parlance, also termed "Diagnosis Keys") when a person is detected as being infected with Covid-19, (ii) to download the published keys for infected people to allow people to check whether they have been near to an infected person and (iii) to download updates to app configuration settings. In addition, some client apps send telemetry data to backend servers about how people have been using the client app e.g. time spent on different screens within the app.

The second component of all of these apps is the Google/Apple Exposure Notification service, that on Android devices is managed by Google and is part of Google Play Services [14].

The Exposure Notification service manages transmission and reception of Bluetooth LE beacons and recording of the duration and signal strength of received beacons [14], [15]. It therefore plays a central role in the contact tracing functionality of the apps.

In summary, we find that the health authority client apps are generally well behaved from a privacy point of view. Indeed, the CoronaWarn (Germany), SwissCovid (Switzerland), StoppCorona (Austria) and Immuni (Italy) apps all appear exemplary with regard to privacy. SmitteStop (Denmark) and RadarCOVID (Spain) also behave well but have the deficiencies that they do not use SSL certificate pinning to verify that they are securely communicating with the correct server, and also that they are closed-source[1]. We recommend that SmitteStop and RadarCOVID both implement certificate pinning and make their app code open source.

Since the authors are Irish we obviously have a special interest in the CovidTracker (Ireland) app. This app makes an initial call to Google's Firebase service but then makes no further use of the service. We have confirmed with the Irish Health Service Executive (HSE) that this call is a hangover from an earlier version of the app and will be removed[2]. CovidTracker sends an "Authorization" HTTP header field in almost all of its communications with the HSE backend server that allows requests from the same handset to be linked together. The Data Protection Impact Assessment for the CovidTracker app [17] states that "IP addresses of users are never transmitted from the networking layer to the backend servers" and so linked requests are not used to track user location over time via the IP address. Nevertheless, such linking of requests over time seems undesirable and is also not clearly communicated by the existing app documentation. We therefore recommend that the "Authorization" HTTP header field be removed, certainly for TEK downloads, but also for "metrics" and "checkins". CovidTracker encourages users to opt-in to sending metrics. These metrics include a mix of device operations data (e.g. number of active users) and medical-related data (e.g. number of close contact notifications, whether a user is diagnosed with Covid-19). We recommend that that these two types of data be held in separate security contexts, e.g. by encrypting medical data so that only medical staff can decrypt it. CovidTracker also uses Google's SafetyNet service to verify handset integrity. This involves the app sending data to Google servers, including the handset hardware serial number, a long-lived identifier of the handset. We recommend that use of the SafetyNet service be discontinued to remove this data sharing with Google and bring CovidTracker into line with other European apps. We discussed these issues with the HSE prior to publication. They acknowledge the issues described here and have said they will work to address them (although of course they may not agree with all of our recommendations).

The ProteGO Safe (Poland) app uses Google's Firebase service to deliver app configuration settings and Google's SafetyNet service to check handset integrity. This means that there are at least two parties involved in handling data, namely Google (who operate the Firebase and SafetyNet infrastructure) and the health authority operating the client app. ProteGO Safe already runs its own server to publish the TEKs of infected people, and so it would presumably be straightforward to also use this to deliver configuration settings, thereby avoiding use of Firebase. As already noted, use of Google's SafetyNet service also seems unnecessary and undesirable. We also observe that ProteGO Safe fails to enforce SafetyNet checks.

The Apturi Covid (Latvia) app uses Firebase Analytics for tracking user interactions with the client app, and this is immediately more intrusive than the other apps we study. As an example of this intrusiveness, a person who is considering taking a COVID-19 test is presumably more likely to linger on screens with relevant information, and that information ought not be exposed to Google. We recommend that the app be changed to allow users to opt in to this tracking, i.e disabling it by default.

In contrast to the client app component, our measurements indicate that the Google Play Services component of these contact tracings apps is troubling from a privacy point of view. Google Play Services connects to Google servers roughly every 20 minutes. These requests necessarily disclose the handset IP address to Google, a rough proxy for location, and also contain persistent identifiers that allow requests from the same device to be linked together. These requests therefore potentially allow fine-grained tracking by Google of device location over time. We note that Google's privacy policy makes clear that it uses IP addresses to estimate location, see the "Your location information" section on the Google Privacy Policy document [18] and it is also stated explicitly in the Firebase (which is part of Google Play Services) documentation that "Analytics derives location data from users' IP addresses" [19]. However, it is not clear for which requests the IP addresses are used in this way and we have asked Google to clarify.

When the "Usage & diagnostics" option in Google Play Services is enabled (which it is by default), then telemetry data on GAEN operation is shared with Google.

The data that Google Play Services sends to Google in these connections also includes, amongst other things, the phone IMEI, the handset hardware serial number, the SIM serial number, the handset phone number, the WiFi MAC address and the user email address. When combined with the potential for fine-grained location tracking via IP address made possible

[1]CoronaWarn, SwissCovis, StoppCorona and Immuni are all open source and use certificate pinning. The SmitteStop developers have confirmed that the app does not use certificate pinning. We note that RadarCOVID is only being piloted just now and so pinning may well have been omitted to simplify testing and will be included in the production version. Absence of certificate pinning means that the transactions of users in, for example, an enterprise network using "Android work" [16] or similar are vulnerable to being exposed to the employer. For example, the act of uploading keys following a positive test phone call may be logged by the employer's network security devices. We note also the RadarCOVID appears to be based upon the D3PT SDK. While this is open source RadarCOVID itself is currently closed-source.

[2]Unlike other open source contact tracing apps, the HSE's github repository does yet not allow issues to be raised and discussed. We asked the HSE to enable issue tracking on 26th June but to date issue tracking remains disabled, though the HSE have recently informed us they plan to remedy this situation.

by the frequent nature of the requests Google Play Services makes to Google servers, on the face of it it is hard to imagine a more intrusive data collection setup.

In our tests we tried to configure the handset so that the minimum possible data is shared with Google when using the Google/Apple Exposure Notification service (we disabled all Google apps and switched off the Google-related settings that we could find, apart from the Exposure Notification service itself). Our measurements of data shared with Google while using these contact tracing apps are therefore an attempt at characterising a minimal baseline and additional data may well be shared in practice if less restrictive Google settings are used. Since this data collection seems to be enabled simply by enabling Google Play Services[3], even when all other Google services and settings are disabled, it is currently an unavoidable aspect of GAEN-based contact tracing apps.

We shared a draft version of this report with Google who responded that their telemetry is "an industry practice" and is explained on Android support pages[4] that also describe ways in which parts of the telemetry can be turned off via a "Usage & diagnostics" setting. We collected measurements with the "Usage & diagnostics" setting both "on" and "off" and observed a broadly similar pattern of network connections, although the content of some of the connections that Google Play Services makes to Google changes, in particular setting this option to "off" disables sharing of GAEN telemetry data with Google.

The broad nature of this data collection and the inability of users to change Google settings so as to opt out of it appears, on the face of it, in conflict with GDPR rules in Europe and we therefore recommend it be brought to the attention of the national data protection authorities. It also indicates that some caution is warranted by governments and health authorities currently promoting widespread use of their contact tracing apps. Concerns regarding the efficacy of the Bluetooth-based contact tracing technology used by GAEN are also pertinent, e.g. see [20], [21], [22], [23], [24], since these indicate that the benefits gained from such sacrifices of privacy remain far from clear.

We make three main recommendations regarding Google Play Services. Firstly, we recommend that user-friendly documentation is made available as to how to make Google Play Services behave in as privacy-friendly a manner as possible while using a GAEN app. Secondly, we recommend that Google implement a "quiet mode" mechanism that enables users who find the Google Play Services data sharing with Google problematic, to easily turn that off. Thirdly, we recommend that the governance of the overall GAEN system be revisited. The health authority client app component has generally received considerable public scrutiny, with publication of detailed Data Protection Impact Assessment documents etc, and we recommend that a governance arrangement that imposes a similar level of scrutiny over the Google/Apple component of the GAEN system is urgently needed.

It is worth noting that there is an ethical quandary in reporting on the type of work we carry out here. The contact tracing apps studied here are deployed and in active use. There is still insufficient data to allow the effectiveness of these apps at protecting public health to be determined, and good reasons to have doubts about their likely effectiveness. However, if the apps are effective and privacy concerns cause people to stop using the apps then that is a potential harm. Equally, since these apps are widely deployed and being used by millions of people across Europe, and privacy concerns have featured prominently in pre-deployment public discussions, concealing new knowledge on the privacy impacts of these apps would be unethical. There is no precedent, as far as we are aware, establishing best practice for responsible disclosure in such a situation. We have informed Google of our findings and delayed publication to allow them to respond. We have also informed the Irish HSE of our findings regarding the CovidTracker app and delayed publication to allow them time to respond, and similarly the developers of SmitteStop, Apturi Covid and ProteGO Safe. A key consideration is what mitigations are possible, and on what time scale can they be deployed. It seems likely that any changes to Google Play Services, even if they were agreed upon, will take a considerable time to deploy (months rather than weeks) and keeping app users in the dark for a long open-ended period seems incorrect.

Regarding shorter-term mitigations, handset users themselves are not powerless. They can, for example, install a firewall on their handset and configure it to block inappropriate connections by Google Play Services. While such firewalls are not available on Google Play Store, open source firewalls such as Blokada [25] and NetGuard can be obtained from app stores such as F-Droid [26]. More work is, however, needed to confirm the compatibility of such firewalls with contact tracing apps.

II. Threat Model: What Do We Mean By Privacy?

It is important to note that transmission of user data to backend servers is not intrinsically a privacy intrusion. For example, it can be useful to share details of the user device model/version and the locale/country of the device and this carries few privacy risks if this data is common to many users since the data itself cannot then be easily linked back to a specific user [27], [28].

Issues arise, however, when data can be tied to a specific user. One common way that this can happen is when an app generates a long randomised string when first installed/started and then transmits this alongside other data. The randomised string then acts as an identifier of the app instance (since no other apps share the same string value) and when the same identifier is used across multiple transmissions it allows these transmissions to be tied together across time.

[3]We contacted Google asking if there was a way to opt-out of this data collection other than by disabling Google Play Services. Google confirmed there was not.

[4]https://support.google.com/android/answer/9021432?hl=en and https://support.google.com/accounts/answer/6078260?hl=en accessed July 16th 2020.

Linking a sequence of transmissions to an app instance does not explicitly reveal the user's real-world identity. However, the data can often be readily de-anonymised. One way that this can occur is if the app directly asks for user details (e.g. phone number, address). But it can also occur indirectly using the fact that transmissions by an app always include the IP address of the user device (or more likely of an upstream NAT gateway). The IP address acts as a rough proxy for user location via existing geoIP services and many studies have shown that location data linked over time can be used to de-anonymise this is unsurprising since, for example, knowledge of the work and home locations of a user can be inferred from such location data (based on where the user mostly spends time during the day and evening), and when combined with other data this information can quickly become quite revealing [29], [30]. A pertinent factor here is the frequency with which updates are sent e.g. logging an IP address location once a day has much less potential to be revealing than logging one every few minutes.

With these concerns in mind, two of the main questions that we try to answer in the present study are (i) What explicit identifying data does each app directly send to its backend servers and (ii) Does the data that each app transmits to backend servers potentially allow tracking of the IP address of the app instance over time.

III. Measurement Setup

A. Viewing Content Of Encrypted Web Connections

All of the network connections we are interested in are encrypted. To inspect the content of a connection we route handset traffic via a WiFi access point (AP) that we control. We configure this AP to use mitmdump [31] as a proxy and adjust the firewall settings to redirect all WiFi traffic to mitmdump so that the proxying is transparent to the handset. In brief, when any app on the handset starts a new web connection the mitmdump proxy pretends to be the destination server and presents a fake certificate for the target server. This allows mitmdump to decrypt the traffic. It then creates an onward connection to the actual target server and acts as an intermediary relaying requests and their replies between the app and the target server while logging the traffic. The setup is illustrated schematically in Figure 1.

The immediate difficulty encountered when using this setup is that all modern apps carry out checks on the authenticity of server certificates received when starting a new connection and aborts the connection when these checks fail. To circumvent these checks we use a rooted phone and use Frida [32] to patch each contact tracing app and Google Play Services on the fly to replace the relevant certificate validation functions with dummy functions that always report validation checks as being passed.

Implementing these changes is, however, a fairly laborious manual process. The bulk of the effort needed lies in deducing how to carry out this patching for Google Play Services as it is closed-source and obfuscated (decompiling the bytecode produces Java with randomised class and variable names etc) plus it uses customised certificate checking code (so standard unpinning methods fail). Many of the contact tracing apps use standard libraries and so can be patched more easily, although we found it slightly harder to patch the Immuni app since it uses a deprecated API and the StoppCorona app since it uses an obfuscated version of the Okhttp3 client.

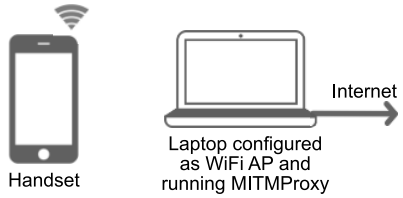


Fig. 1. Measurement setup. The mobile handset is configured to access the internet using a WiFi access point hosted on a laptop, use of cellular/mobile data is disabled. The laptop also has a wired internet connection. When an app on the handset starts a new web connection the laptop pretends to be the destination server so that it can decrypt the traffic. It then creates an onward connection to the actual target server and acts as an intermediary relaying requests and their replies between the handset app and the target server while logging the traffic.

B. Connection Data

Since the content of connections is not especially human-friendly they are summarised and annotated in Appendix 2. The raw connection data is also available on request by sending an email to the author (since it contains identifiers and authorization keys, posting the raw data publicly is probably unwise).

C. Hardware and Software Used

Mobile handset: Google Pixel 2 running Android 9 with Google Play Services 20.24.14 and Google Exposure Notification Service v202606000. Rooted using Magisk v19.1 and Magisk Manager v7.1.2 and running Frida Server v12.5.2. Contact tracing app versions used: CoronaWarnApp v1.0.4, SwissCovid v1.0.5, ApturiCovid 1.0.47, ProteGO Safe v4.2, CovidTracker v1.0.40, SmitteStop v1.0.2, RadarCOVID v1.0.0. Laptop: Apple Macbook running Mojav 10.14.6 running Frida 12.8.20 and mitmproxy v5.0.1. Using a USB ethernet adapter the laptop is connected to a cable modem and so to the internet. The laptop is configured using its built in Internet Sharing function to operate as a WiFi AP that routes wireless traffic over the wired connection. The laptop firewall is then configured to redirect received WiFi traffic to mitmproxy listening on port 8080 by adding the rule `rdr pass on bridge100 inet proto tcp to any port 80, 443 -> 127.0.0.1 port 8080`. The handset is also connected to the laptop over USB and this is used as a control channel (no data traffic is routed over this connection) to carry out dynamic patching using Frida. Namely, using the adb shell the Frida server running on the handset is controlled from the laptop. In the Developer Options screen on the handset the "Stay Awake" option is set on.

D. Google Device Settings

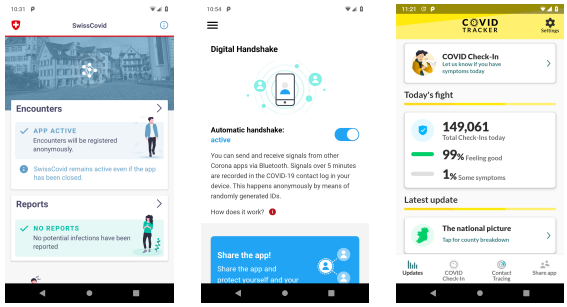
While Android was developed by Google, it is open source and can be used without interacting with Google services e.g. this is common in China due to restrictions on use of Google services there. Google Play Services is a different matter. Google Play Services is a closed-source proprietary app that provides Google services used by other apps, including analytics, crash reporting, messaging, advertising, fused location and so on. On Android the Google/Apple Exposure Notification service is implemented within Google Play Services and so Google Play Services *must* be enabled in order to use the apps, there is no workaround to this.

In our tests we tried to configure the handset so that the minimum possible data is shared with Google when using the Exposure Notification service. Unless otherwise stated we therefore disabled all Google apps on the handset (Chrome, Drive, Gmail, Google, Google Play Movies, Google Play Music, Google Play Store, Maps, Photos, YouTube) and Google services (Actions Services, ARCore, Google Connectivity Services, Google Support Services, Google Text-to-Speech Engine, Google VR Services, Pixel Visual Core Services, Project Fi). In the Settings→Google screen (which only appears when Google Play Services is enabled) we also disabled all 'Account Services (Connected App, Contact Sync, Google Fit, Google Play Instant, Google Pay), Ad settings (Opt out of ads personalisation, Enable debug logging for ads), Autofill (Autofill with Google, SMS verification), Backup to Google Drive, Data & Messaging (App preview messages), Device Connections (Android Auto, Cast media controls), Parental Controls, Security (Find My Device, Google Play Protect) and Firebase App Indexing. The Covid-19 Exposure Notification service is, of course, left enabled. On the Settings→Google screen at the top right corner are three dots, which when clicked open a menu with "Usage & diagnostics" and "Open source licences". Clicking on "Usage & diagnostics" opens a screen with information on telemetry plus a toggle switch. Although not hidden, this switch is far from prominent and in fact we missed it on our initial passes through the Google setting screen. Since the switch is on by default, and is not so easy to find, we collected measurements both with it set on and set off. A fresh gmail account is used to register the device, which is not whitelisted for use with GAEN.

Our measurements of data shared with Google while using the contact tracing apps are therefore an attempt at characterising a minimal baseline. Additional data may well be shared in practice if less restrictive Google settings are used.

E. Test Design

Test design is straightforward since the all of the apps support only a single flow of user interaction. Namely, in all of the apps there is an "onboarding" process that involves stepping through a sequence of screens until the main screen is reached, see Appendix A for details. Once the main screen is arrived at this is displayed thereafter, see Figure 2 for some examples of main screens. This main screen has typically has buttons for help, sharing of the app, plus a button that is only to be pressed when the user has been confirmed as infected with Covid-19.



(a) SwissCovid (b) StoppCorona (c) CovidTracker

Fig. 2. Main screens displayed by three of the contact tracing apps following initial setup.

Testing therefore consists of recording the data sent upon installation and startup of the app, followed by navigation through these screens until the main screen is reached. The data sent by the app when left idle at the main screen (likely the main mode of operation of the app) is also recorded. We did not investigate the data sent upon pressing the upload function since we did not want to risk interfering with operation of the live contact tracing service. A number of apps also have symptom checking functions and we also left these untouched to avoid interfering with the live service.

F. Finding Identifiers In Network Connections

Potential identifiers in network connections were extracted by manual inspection. Basically any value present in network messages that stays the same across messages is flagged as a potential identifier. As we will see, many of the values of interest are associated with Google Play Services. We therefore try to find more information on the nature of observed values from Google privacy policies and other public documents as well as by comparing them against known software and device identifiers e.g. the IMEI, hardware serial number, Google advertising identifier of the handset.

IV. Google Firebase

ProteGO Safe and Apturi Covid both use Google's Firebase service, which is part of Google Play Services. This means that there are at least two parties involved in handling data shared by the app, namely Google (who operate the Firebase service infrastructure) and the health authority (or other agency) operating the client app. As owner of Firebase, Google has access to all data transmitted by the app via Firebase but filters what data is made available to the operator of the app e.g. to present only aggregate statistics [19].

ProteGO Safe makes use of Firebase Remote Configuration while Apturi Covid makes use of Firebase Analytics (also referred to as Google Analytics for Firebase).

The Firebase privacy documentation [33] outlines some of the information that is exchanged with Google during operation of the API. Firebase Analytics makes use of a number of

identifiers including: (i) a user-resettable Mobile ad ID to "allow developers and marketers to track activity for advertising purposes. They're also used to enhance serving and targeting capabilities." [34], (ii) an Android ID which is "a 64-bit number (expressed as a hexadecimal string), unique to each combination of app-signing key, user, and device" [35], (iii) an InstanceID that "provides a unique identifier for each app instance" and "Once an Instance ID is generated, the library periodically sends information about the application and the device where it's running to the Firebase backend. " [36] and (iv) an Analytics App Instance ID that is "used to compute user metrics throughout Analytics" [19]. The Firebase Analytics documentation [19] states that "As long as you use the Firebase SDK, you don't need to write any additional code to collect a number of user properties automatically", including Age, Gender, Interests, Language,Country plus a variety of device information. It also states that "Analytics derives location data from users' IP addresses".

The data collected by Google during operation of its Firebase services need not be stored in the same country as the user of an app is located. The Firebase privacy documentation [33] states that "Unless a service or feature offers data location selection, Firebase may process and store your data anywhere Google or its agents maintain facilities".

V. Data Transmitted By Client Apps

We begin by looking at the data shared by the client component of the contact tracings apps, i.e. the component managed by the public health authority of each country. The detailed connection measurements are given in Appendix 2 and are only summarised here.

A. CoronaWarn (Germany), SwissCovid (Switzerland), Immuni (Italy), StoppCorona (Austria), RadarCOVID (Spain), SmitteStop (Denmark)

We treat these six apps together since they exhibit very similar behaviour. In summary, they make a minimal number of requests to back end servers and use no persistent identifiers. Arguably, from a privacy viewpoint they therefore represent best practice in the design of such client apps. We note that SmitteStop and RadarCOVID are closed source while the other apps are all open source [5], [7], [6], [8], and many of the latter have benefited from public discussion via issue trackers on github that identify potential improvements and ways to fix deficiencies.

1) Data Sent On Initial Startup: With the exception of SmitteStop, on first startup all of these apps connect to national backend servers to download configuration settings and the published exposure keys of infected people. The requests contain no cookies or other identifiers linking them to the app instance. StoppCorona requests do contain an authorizationKey header but its value is the hard-wired into the app and is the same for all instances, it therefore does not act to identify a specific instance of the app. RadarCOVID downloads a UUID value on start up but we did not observe this being sent back to the server in subsequent requests (we speculate that it may be used when uploading keys upon a positive diagnosis but, as already noted, we did not check that functionality to avoid interfering with operation of the live app).

2) Data Sent When Sitting Idle At Main Screen: When left idle all of these apps periodically connect to national backend servers to check for updates to their configuration settings and the published exposure keys. These requests contain no app instance identifiers[5]. SmitteStop also uploads log event data to app.smittestop.dk/API/v1/logging/logMessages. We observed this data to include stack traces and other information relating to the operation of the app, but did not see evidence of identifiers related to the app instance. Requests are made relatively infrequently, no more than about 4 times a day.

B. ProteGO Safe (Poland)

ProteGO Safe makes use of the Google Firebase service to store configuration settings, it also makes use of Google's SafetyNet service. This means that there are at least two parties involved in handling data shared by the app, namely Google (who operate the Firebase service infrastructure) and the health authority (or other agency) operating the client app. As owner of Firebase, Google has access to all data transmitted by the app via Firebase but filters what data is made available to the operator of the app e.g. to present only aggregate statistics [19]. ProteGO Safe is open source [10].

We note that the use of Firebase by ProteGO Safe to store configuration is perplexing. ProteGO Safe already uses a separate server at exp.safesafe.app to publish keys of infected people, so it would seem to be trivial to also use that server to publish configuration settings and so avoid sharing of data with Firebase. ProteGO Safe's use of SafetyNet also seems problematic. Firstly, ProteGO Safe does not respond to the failure of our handset to pass the SafetyNet checks and secondly checks are verified within the app itself, which is not best practice since they can then be overridden (using similar techniques to those we used to bypass certificate checks on encrypted web traffic). Since SafetyNet use in the app is therefore largely ineffective, it would be simpler to just avoid using the service and thus avoid sending data to Google.

1) Data Sent On Initial Startup: ProteGO Safe initialises Firebase upon first startup, which generates the following POST request (standard/uninteresting parameters/headers are omitted) :

```
POST https://firebaseinstallations.googleapis.com/v1/
projects/safesafe-app/installations
Headers:
      X-Android-Package: pl.gov.mc.protegosafe
Request body:
      "fid": "cY46N0YUR_ykuI0m_Bx-kz",
```

The "fid" value is the Firebase Instance ID. This uniquely identifies the current instance of the app. The response to this request echoes the fid value and includes two tokens which

[5]Requests made by SmitteStop contain an "Authorization_Mobile" but its value is hard-wired in the app and so is the same for all instances. SmitteStop requests for updated configuration settings also send a cookie. The cookie value appears, however, to be the same for all instances of the app. One might speculate that these values originally changed per app instance but a decision was taken later to fix them to be the same for all instances.

appear to be used to identify the current session. In a second POST request ProteGO Safe sends the fid to android.clients.google.com together with a persistent device identifier (likely the androidID, which is set on first startup of a device and only changes upon a factory reset).

ProteGO Safe now makes two calls to Google's SafetyNet service. The first call sends data that includes the handset hardware serial number HT85G1A05... (a persistent identifier of the handset):

```
POST https://www.googleapis.com/androidantiabuse/v1/x/create
Headers:
  User-Agent: DroidGuard/202117028
Request body:
  <...>
   \x06SERIAL\x12\x0cHT85G1A05...\x12\x14
   <...>
```

Note that our patched version of Google Play Services fails the SafetyNet checks, but ProteGO Safe does not respond to this failure. ProteGO Safe also carries out SafetyNet checks within the app itself, where they can be overridden.

ProteGO Safe now fetches configuration information from Firebase using:

```
POST https://firebaseremoteconfig.googleapis.com/v1/
projects/466787798978/namespaces/firebase:fetch
Request body:
    <...>
   "appInstanceId": "cY46N0YUR_ykuI0m_Bx-kz",
    "appInstanceIdToken": "
cY46N0YUR_ykuI0m_Bx-kz:APA91...c",
```

The "appInstanceId" is a persistent identifier linked to the specific instance of the app, the "appInstanceIdToken" is a value returned by Firebase in response to earlier requests and so can be linked to the fid and device and also acts to link together requests made to Firebase.

ProteGO Safe makes a call to exp.safesafe.app to fetch details of published keys. No persistent identifiers are sent with this request.

2) Data Sent When Sitting Idle At Main Screen: Several times a day ProteGO Safe fetches configuration information from Firebase using similar requests as above, these include the appInstanceId and appInstanceIdToken headers. Less frequently it also makes the call to exp.safesafe.app to fetch details of published keys.

C. Apturi Covid (Latvia)

Apturi Covid makes use of the Google Firebase Analytics service. This means that it shares data with Google servers both on initial startup and afterwards while idle and that it tracks user interactions with the app. There is no option to opt out of this tracking. Apturi Covid is closed source.

1) Data Sent On Initial Startup: Similarly to ProteGO Safe, on first startup Apturi Covid initialises Firebase and sends an fid value to Google that uniquely identifies the current instance of the app. It then sends a persistent device identifier along with the fid value to android.clients.google.com.

Apturi Covid now makes its first call to Firebase Analytics:

```
POST https://app-measurement.com/a
  <...>
    \x02_o\x12\x04auto
   \x17
   \x04_cis\x12\x0freferrer API v2\x12\x04_cmp\x18\x96\x98\
xe6\xd9\xb1. \x00\x1a\x14\x08\xcd\xde\xe5\xd9\xb1.\x12\
x04_fot \x80\x8c\xa8\xdb\xb1.\x1a\x0e\x08\xcd\xde\xe5\xd9\
xb1.\x12\x03_fi \x01\x1a\x0f\x08\xac\xdf\xe5\xd9\xb1.\x12\
x04_sno \x01\x1a\x13\x08\xac\xdf\xe5\xd9\xb1.\x12\x04_sid \
xcf\xa6\x83\xf8\x05\x1a\x0f\x08\xb8\xe8\xe6\xd9\xb1.\x12\
x04_lte \x01\x1a\x0e\x08\xb8\xe8\xe6\xd9\xb1.\x12\x03_se \
x00 \xa4\xe8\xe6\xd9\xb1.(\xac\xdf\xe5\xd9\xb1.0\x96\x98\xe6
\xd9\xb1.8\xcd\xde\xe5\xd9\xb1.B\x07androidJ\x019R\x07Pixel
2Z\x05en-us`<r\x17lv.spkc.gov.apturicovid\x82\x01\x061.0.47\
x88\x01\xe0\xda\x01\x90\x01\x85\xab\x0c\x9a\x01$
1d2635f5-2af7-4fb3-86e8-5fd6...\xa0\x01\x00\xaa\x01
f5fd7266cb7df65bdf87d6788d550c3e\xb0\x01\x84\x87\xdc\xaa\xf9
\xa3\xeb\xf6\xaa\x01\xb8\x01\x02\xca\x01-1:503076402814:
android:5d8cba16d971f6221aca2f\xd0\x01\xcd\xde\xe5\xd9\xb1.\
xe0\x01\x01\xf2\x01\x16c8WD_GfwSvWv89aGYrWd9d\xf8\x01/\x98\
x02\xec\xae\x99\x93\x80\x84\xea\x02\xe8\x02\xc8\xeb\x86\x0b\
xf0\x02
```

The payload includes the Google advertising id of the device (1d2635f5-2af7-4fb3-86e8-5fd6...), the app_instance_id (f5fd7266cb7df65bdf87d6788d550c3e) and the fid (c8WD_GfwSvWv89aGYrWd9d). Unless manually reset by the user the value of the Google advertising id persists indefinitely, including across fresh installs of Apturi Covid, and so essentially acts as a strong identifier of the device and its user.

Apturi Covid now makes calls to apturicovid-files.spkc.gov.lv to fetch configuration settings, check for published keys and fetch infection statistics. These requests do not contain identifiers.

2) Data Sent When Sitting Idle At Main Screen: While idle Apturi Covid makes fairly frequent calls to Firebase Analytics to log data on user interactions with the app. As before, these requests include he Google advertising id of the device, the app_instance_id and the fid.

Apturi Covid also makes calls to apturicovid-files.spkc.gov.lv to fetch configuration settings, check for published keys and fetch infection statistics. As before, these requests do not contain identifiers.

D. CovidTracker (Ireland)

CovidTracker makes use of Google's SafetyNet service. On startup it also connects to Google's Firebase service - inspection of the code and discussion with the HSE confirmed this is "path-not-taken" code and will hopefully be removed. The app is open source [12] though the published code does not enable certificate pinning whereas the released app does.

1) Data Sent On Initial Startup: Similarly to ProteGO Safe and Apturi Covid, on first startup (before any user interaction) CovidTracker initialises Firebase and sends an fid value to Google that uniquely identifies the current instance of the app:

```
POST https://firebaseinstallations.googleapis.com/v1/
projects/api-71643941211319615 44-290715/installations
Headers:
   X-Android-Package: com.covidtracker.hse
Request body:
   "fid": "do6qB-2BSDSRi2XLZIr-ul"
```

It then sends a persistent device identifier along with the fid value to android.clients.google.com.

CovidTracker then downloads configuration settings from app.covidtracker.ie. No identifiers are sent with this request.

Once the user navigates past the consent page within the app CovidTracker makes a series of calls to Google's SafetyNet service, similarly to ProteGO Safe (although unlike ProteGO Safe failure to pass the safety checks is not ignored). In these calls the app sends the device hardware serial number to Google:

```
 POST
https://www.googleapis.com/androidantiabuse/v1/x/create
Headers:
   User-Agent: DroidGuard/202117028
Request body:
   <...>
    \x06SERIAL\x12\x0cHT85G1A05...\x12\x14
    <...>
```

At the end of this exchange the app.covidtracker.ie server sends a "refreshToken" and a "token" value to the app. The "token" value (eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI..4k) is then sent in the "Authorization: Bearer" header of subsequent requests and so links these requests together.

Inspection of the app source code indicates that the refreshToken value effectively acts as a persistent identifier of the app instance. It is used to request new token values from the app.covidtracker.ie server, but these token values are themselves linked by the refreshToken value.

Following this exchange CovidTracker checks for published keys for infected people using:

```
GET https://app.covidtracker.ie/api/exposures/
 Headers:
   authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Parameters:
    since: 0
    limit: 1
```

and fetches information on Irish infection statistics:

```
GET https://app.covidtracker.ie/api/stats
Headers:
   authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
```

When the user has agreed to collection of telemetry data the app also sends that to app.covidtracker.ie using calls such as:

```
POST https://app.covidtracker.ie/api/metrics
Headers:
   authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Request body:
    { "event": "CALLBACK_OPTIN",
      "os": "android",
      "version": "1.0.40"}
```

The Data Protection Impact Assessment for the CovidTracker app [17] states at the start of Section 4.3 that "IP addresses of users are never transmitted from the networking layer to the backend servers thus minimising the possibility of inadvertently recombining IP address and payload data". Hence, while requests are linked together by the "Authorization: Bearer" header value that is sent with them this is not used to track user location over time via the IP address. Nevertheless, while such linking of requests may well be useful for symptom checking functionality and the like, it seems unnecessary to also link together the requests of users who have not opted in to this. In particular, it seems unnecessary to link the requests to app.covidtracker.ie/api/exposures/ used to check for new published keys to each other and to other requests made by the app.

2) Data Sent When Sitting Idle At Main Screen: Roughly every two hours CovidTracker checks for updates to the published keys for infected people using:

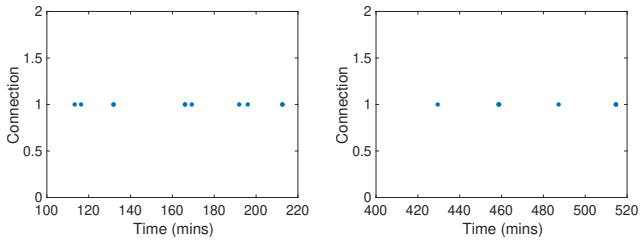
```
GET https://app.covidtracker.ie/api/exposures/
 Headers:
   authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Parameters:
    since: 0
    limit: 1
```

E. Certificate Pinning, DNSSEC & Mail

SSL certificate pinning allows apps to verify that they are securely communicating with the correct server. Pinning involves hardwiring server certificate details within the app, allowing the app to confirm that the certificate presented to it by a server is as expected and is not, for example, being spoofed by a man-in-the-middle attacker. We verified that certificate pinning is implemented by CoronaWarn, SwissCovid, StoppCorona, Immuni, ProteGO Safe, Apturi Covid and CovidTracker. Certificate pinning is not, however, implemented by SmitteStop or RadarCOVID, but we note that RadarCOVID is only being being used in a pilot trial and the server domain names being used also seem like they may be temporary ones; we presume that pinning would be used in the production version. The privacy impact of an absence of pinning is that the transactions of users in, for example, an enterprise network using "Android work" [16] or similar are vulnerable to being exposed to the employer. The net effect for example could that the act of uploading keys following a positive test phone call may be logged by the employer's network security devices.

DNSSEC is a standard data integrity mechanism that makes attacks against the DNS service used to map from server names to IP addresses much harder. Of all the DNS names mentioned above, only two (exp.safesafe.app used by ProtegGO Safe and www.pt.bfs.admin.ch used by SwissCovid) are DNSSEC-signed. Given the sensitivity and scale ambitions for contact tracing apps, and the fact that deploying DNSSEC is not complex, we would have expected public health authorities to have deployed all cost-effective security measures.

We also looked at the set of mail addresses that were hard-coded into these apps or part of a downloadable configuration. We assume these mail addresses provide users a means for further contact and so are liable to receive messages with sensitive content. The use and setup for transport layer security (STARTTLS) for the relevant mail servers is therefore also of interest. Not all apps appear to encourage use of email though of course there may be email addresses on web pages that we did not see. For those we found, none of the mail servers support MTA-STS [37] and only one (rki.de) appears to support DANE. [38] Of the rest, one (Italian, sanita.it) has



(a) (b)

Fig. 3. Typical time histories of network connections made by Google Play Services when (a) "Usage & diagnostics" option is on, and (b) when it is off.

no support for TLS at all. Of the two other Italian mail servers, one (postacert.sanita.it) uses a certificate that may be self-signed or signed by some local CA, another (immuni.italia.it) uses a ciphersuite that is no longer recommended (DHE-RSA-AES256-GCM-SHA384) and uses that with a too-short Diffie-Hellman value (1024 bits). The Irish HSE mail servers appear to use self-signed certificates for TLS and hence are vulnerable to active MX spoofing attacks [39]. One of the domains associated with the Polish app (mc.gov.pl) also uses a self-signed certificate, whereas another (gis.gov.pl) appears to have a clean setup. The German and Austrian apps seem to be associated with mail addresses where the servers have a clean-looking setup. Overall we can see that, as is often the case with web applications, the linkage back to email security has not been fully considered in many services.

VI. Data Transmitted By Google Play Services

As already noted, all of the contact tracing apps are formed from two components, the client app and the Google/Apple Exposure Notification service, that on Android devices is managed by Google and is part of Google Play Services.

We tried to configure the handset so that the minimum possible data is shared with Google when using the Exposure Notification service, see Section III-D, and so our measurements of data shared with Google should therefore be thought of as a minimal baseline. To the best of our knowledge there is no way to disable these connections other than by disabling Google Play Services itself (Google have subsequently confirmed this), but that would then prevent use of a GAEN-based contact tracing app. The detailed connection measurements are given in Appendix 2 and are only summarised here.

Figure 3 shows typical time histories of network connections made by Google Play Services[6]. When the "Usage & diagnostics" option is on the mean time between connections is 17.5 minutes, and when it is off the mean time between connections increases slightly to 25 minutes. We also collected measurements with the Google Play Store is enabled as well as Google Play Services. We observed that the overall rate of network connections is much the same but that Google Play Services makes somewhat fewer connections, it seems that Google Play Services and Google Play Store therefore may co-operate in some way to regulate the number of network connections they jointly make.

The fact that Google Play Services connects to Google servers so frequently is pertinent because every connection to a Google server necessarily reveals the handset IP address. As already noted, the handset IP address is a rough proxy for location and so making connections every 20 minutes or so potentially allows Google to track the handset location in quite a fine-grained manner.

This is not a hypothetical concern. Google's privacy policy makes clear that it uses IP addresses to estimate location, see the "Your location information" section on the Google Privacy Policy document [18] and this is also stated explicit in the Firebase documentation [19] which states that "As long as you use the Firebase SDK, you don't need to write any additional code to collect a number of user properties automatically", including Age, Gender, Interests, Language,Country plus a variety of device information, and that "Analytics derives location data from users' IP addresses".

A. Connections to play.googleapis.com/p/log/batch

Google Play Services sends data to Google by making requests to https://play.googleapis.com/p/log/batch several times an hour. Each request is of the form:

```
POST https://play.googleapis.com/p/log/batch
Headers:
   Authorization: Bearer ya29.a0AfH6SMCv...
   X-SERVER-TOKEN: CAESKQDyi0h8u1NFMSbtIY...
   Cookie: NID=204=cZfpA_V3EeVcgH8ON4DUR...
   User-Agent: Android/com.google.android.gms/202117028 (
walleye PPR2.180905.005); gzip
```

The cookie, Authorization and X-SERVER-TOKEN headers sent with this request all contain persistent identifiers that can be used to link requests together and to the specific handset.

We observe a wide range of data being sent as the payload to these requests, the payload often being quite large. This includes information and telemetry on apps and services installed on the handset, fine-grained details of errors logged (including execution backtraces), details of network connection errors etc. In addition to the cookie, Authorization and X-SERVER-TOKEN headers we also observed the handset phone number (+353892197...) and the SIM serial number (8935311180135555..) being sent:

```
\x80\xd3\x99\xfa\xb0.*\xa5\x1e\xee\x02\x08\x9f\xd6\xbe\xf7\
xb0.2\x7f\x12\x00\x1a(\x08\x88\xa0\xd5\x1c\x12 6.0.117 (
Xorn_RC10.phone_dynamic)"\x02
   \x00\xb2\x01I
   G\x08\x01\x10\x00\x18\x02"\x0527211*\x0c\x08\x01\x10\x02\
x1a\x06\x08\x01\x10\x01\x18\x012\x138935311180135555...:\x13
8935311180135555...H\x01\xd0\x01\x03X\x0ex\xa08\x88\x01\x85\
xec\xa5\x1d\x82\x01\xcd\x01\x12\xac\x01\x08\x03\x12\xa7\x
<...>
\xa3\xa4\xfa\x80\xb4.*\xbc[\x80\x03\x08\xed\xef\xea\xfd\xb3
.2\x8e\x01\x12\x00\x1a(\x08\xa8\x88\x91\x1d\x126.1.097 (
Yeti_RC11.phone_dynamic)"\x0f

   +353892197...\xb2\x01K
```

[6]Each network connection is linked to the app which made it by running the netstat command on the handset over an adb shell, and confirmed by inspection of the connection contents which typically contain either a "User-Agent" header or an "app" that indicates the app.

When the"Usage & diagnostics" option is set on then telemetry relating to the Exposure Notification service is also sent[7], see Appendix 2.

B. Connections to android.googleapis.com/checkin

A few times per day Google Play Services sends data to Google by making a request to https://android.googleapis.com/checkin:

```
POST https://android.googleapis.com/checkin
Headers:
   Cookie: NID=204=Z-_RXuS4ZdrKlKkBoa...
   User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; Pixel 2
Build/PPR2.180905.005)
   X-SERVER-TOKEN: CAESKQDyi0h8u1NFMSbtIY...
Request body:
<...>
2018-09-05\x10\xe8\xc3\xa7\x9d\xa3.2\x0527205:\x0527211B\
x06WIFI::H\x00p\x02z\x15\x08\x08\x10\x01\x1a\x0bunspecified
"\x00(\x00\x82\x01]
   \x0527211\x12\x0cTesco Mobile\x1a\x010 \x00 \x01 \x022\
x0f272110103800000:(0
AFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB\x02\x15\xe5\x90\x01
\x01\x9a\x01\x04WIFI2\x05en-US8\xfc\xa0\xab\xfb\xd7\xf6\xce\
x8d\xc9\x01J\x0c404e36d3f4bdR\x0f35753708924...Z\x1a[
<...>@gmail.com]Z\xc7\x02ya29.a0AfH6SMCv...
   Europe/Dubliniv\xd4\xa3\x7f\x84\xb0\xf7;p\x03z\
x1cbfMkwynjHzXGBPc2WT62otR8JkI=\x82\x01\x0cHT85G1A05...\x92\
x01\x92\\\x08\x03\x10\x01\x18\x01 <...>
```

A cookie is sent with the request that is the same as that sent with the play.googleapis.com/p/log/batch requests thus allowing them to be linked together. Also sometimes an X-SERVER-TOKEN header value (highlighted in bold above) that matches that sent with the play.googleapis.com/p/log/batch requests. The Authorization: Bearer header value (ya29.a0AfH6SMCv...) sent with play.googleapis.com/p/log/batch is also sometimes included in the request body. In addition, the payload contains the phone IMEI (35753708924...), the hardware serial number (HT85G1A05...), the SIM serial number (not shown in snippet above), the WiFi MAC address (404e36d3f4bd) and the user email address (<...>@gmail.com). These are all long-lived hardware identifiers that do not change between reinstalls of the app or even factory reset of the handset.

Using the data sent in these p/log/batch and checkin requests, Google therefore gathers detailed, fine-grained information on how the handset is being used and can link this data to the handset hardware, SIM and user email. When combined with the fine-grained location tracking via IP address made possible by the frequent nature of the requests Google Play Services makes to Google servers its hard to imagine a more intrusive data collection setup. Recall that as far as we can tell this data collection is enabled simply by installing Google Play Services, even when all other Google services and settings are disabled. It therefore appears to be unavoidable for users of GAEN-based contact tracing apps.

Some may consider that the kind of data gathering by Google described here is nothing new or unexpected and that mobile phone users have already factored in the risks of these kinds of corporate surveillance. We believe there are at least three reasons why this is an unsatisfactory response: a) governments and public health authorities are strongly encouraging their entire population to use these apps, and hence are (wittingly or not) pressurising their entire populations to take part in this corporate surveillance, b) it is highly likely that many users and even app developers are unaware of the detail and the level of intrusiveness described here and c) the lack of an opt out from this data collection seems in conflict with GDPR.

[7]We have confirmed this with Google, who also advise that when the "Usage & diagnostics" option is set off this Exposure Notification telemetry is no longer sent. However, we continued to observe phone identifiers, including the phone number and the SIM serial number, being sent in these requests even when the "Usage & diagnostics" option is set off.

C. Other Connections

Google Play Services also makes a number of other network connections. These include an authorisation request, typically made once or twice a day:

```
POST https://android.googleapis.com/auth
Headers:
   device: 3e7736d127071c1a
   app: com.google.android.gms
   User-Agent: GoogleAuth/1.4 (walleye PPR2.180905.005);
gzip
Request body:
  androidId: 3e7736d127071c1a
   app: com.google.android.gms
   Email: <...>@gmail.com
   client_sig: 38918a453d07199354f8b19af05ec6562ced5788
   Token: aas_et/AKppINaJgeUW3nalnMjiqfI8KVMmfTkO...g9w=
   callerSig: 38918a453d07199354f8b19af05ec6562ced5788
```

Observe that this request sends the Android ID, a persistent id which is unique to each combination of app-signing key, user, and device" [35], see also Section IV. It also sends the user gmail address.

A request is also made to www.googleapis.com/experimentsandconfigs a few times a day. When the "Usage & diagnostics" option is on this sends details of the configuration of the Exposure Notification service.

D. Google Play Store Connections

Unless otherwise stated, in all of our measurements Google Play Store was disabled. However, we also collected some additional measurements with Google Play Store enabled, since we expect that most users will use the Google Play Store to install their contact tracing app (and likely leave it always enabled in any case). Google Play Store can, however, be disabled without interfering with operation of GAEN-based contact tracing apps (although disabling Google Play Store prevents app updates).

We generally observed that the payloads of Google Play Store connections were less intrusive to privacy than the Google Play Services connections noted above. Requests often include an Authorization header than can be used to link them together, and Google Play Store makes requests to android.googleapis.com/auth and www.googleapis.com/experimentsandconfigs that share similar data with Google as those made by Google Play Services, although the www.googleapis.com/experimentsandconfigs include in addition the handset Google Advertising ID (a persistent identifier used for personalised advertising, which can be manually reset by the user but otherwise remains unchanged indefinitely).

E. Recommendations With Respect To Google Play Services

We recognise that Google Play Services is an extremely widely-used software component and so may not be easily modified. It may also be that changes to make Google Play Services more suitable for use with GAEN apps make it less suited for current commercial purposes, however we recommend three potential avenues to help mitigate the intrusiveness documented above.

Firstly, there is a significant lack of documentation as to the internals of the GAEN implementation and of the other parts of Google Play Services needed for apps to operate. In particular, usable documentation as to how to make Google Play Services behave in as privacy-friendly a manner as possible while using a GAEN app should assist public health authorities in supporting informed decision-making for their users and potential users. We are aware of no such documentation. We have been informed that Google and Apple themselves are aware of and plan to address this issue, hopefully in the very near future.

Secondly, Google could implement a "quiet mode" mechanism that enables users who find the Google Play Services data sharing with Google problematic to easily turn that off. That should assist a cohort of privacy-conscious users who might not otherwise adopt GAEN apps.

Thirdly, public health authorities, civil societies and others could, along with Google and Apple, revisit the governance issues associated with the GAEN system overall. While the involvement of Google and Apple seems entirely necessary (for their handset expertise at least), the current situation where Google and Apple are essentially gatekeepers who are in ultimate control of the behaviour of these apps seems undesirable. A governance setup that imposes a similar level of scrutiny over both the client app component and the Google/Apple component of the GAEN system seems sensible and necessary.

VII. Conclusions

We describe the data transmitted to backend servers by the contact tracing apps now deployed in Germany, Italy, Switzerland, Austria, Denmark, Spain, Poland, Latvia and Ireland with a view to evaluating user privacy. These apps consist of two separate components: a "client" app managed by the national public health authority and the Google/Apple Exposure Notification service, that on Android devices is managed by Google and is part of Google Play Services. We find that the health authority client apps are generally well behaved from a privacy point of view, although the privacy of the Irish, the Polish and Latvian apps could be improved. In marked contrast, we find that the Google Play Services component of these apps is extremely troubling from a privacy viewpoint. Google Play Services contacts Google servers roughly every 20 minutes, poentially allowing fine-grained location tracking via IP address. In addition, Google Play services also shares the phone IMEI, hardware serial number, SIM serial number, handset phone number and user email address with Google, together with fine-grained data on the apps running on the phone. This data collection is enabled simply by enabling Google Play Services, even when all other Google services and settings are disabled. It therefore appears to be unavoidable for users of GAEN-based contact tracing apps on Android. This level of intrusiveness seems incompatible with a recommendation for population-wide usage. It also seems incompatible with the following statement from Google: "We understand that the success of this approach depends on people feeling confident that their private information is protected. The Exposure Notifications System was built with your privacy and security central to the design. Your identity is not shared with other users, Google or Apple."[8] We note the health authority client app component of these contact tracing apps has generally received considerable public scrutiny and typically has a Data Protection Impact Assessment, whereas no such public documents exist for the GAEN component of these apps. Extending public governance to the full contact tracing ecosystem, not just of the health authority client app component, therefore seems to be urgently needed if public confidence is to be maintained.

Acknowledgements

Trinity College Dublin, (the authors' employer) funded the "Testing Apps for Contact Tracing" (TACT) project[9] that has allowed us the time and handsets required here.

References

[1] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser, "Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing," *Science*, 2020.

[2] Irish Times, "EU urges vigilance to avoid coronavirus second wave," 17 May 2020. [Online]. Available: https://www.irishtimes.com/news/world/europe/eu-urges-vigilance-to-avoid-coronavirus-second-wave-1.4255632

[3] "Apple and Google partner on COVID-19 contact tracing technology," 10 April, 2020. [Online]. Available: https://www.apple.com/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/

[4] Google Blog, "Exposure Notification API launches to support public health agencies," Accessed 13 June 2020. [Online]. Available: https://blog.google/inside-google/company-announcements/apple-google-exposure-notification-api-launches/

[5] "CoronaWarnApp Source Code," Accessed 9 July 2020. [Online]. Available: https://github.com/corona-warn-app

[6] "Immuni Source Code," Accessed 9 July 2020. [Online]. Available: https://github.com/immuni-app

[7] "SwissCovid Source Code," Accessed 9 July 2020. [Online]. Available: https://github.com/DP-3T/dp3t-app-android-ch

[8] "StoppCorona Source Code," Accessed 9 July 2020. [Online]. Available: https://github.com/austrianredcross/stopp-corona-android

[9] "SmitteStop App," Accessed 12 July 2020. [Online]. Available: https://smittestop.dk/

[10] "ProteGO Safe Source Code," Accessed 9 July 2020. [Online]. Available: https://github.com/ProteGO-Safe/android

[11] "Apturi Covid App," Accessed 9 July 2020. [Online]. Available: https://apturicovid.lv/

[8] https://www.google.com/covid19/exposurenotifications/ accessed 13th July 2020.

[9] See https://down.dsg.cs.tcd.ie/tact/

[12] "CovidTracker Source Code," Accessed 9 July 2020. [Online]. Available: https://github.com/HSEIreland

[13] "Radar COVID App, Google Play Store," Accessed 12 July 2020. [Online]. Available: https://play.google.com/store/apps/details?id=es.gob.radarcovid

[14] "Exposure Notifications: Android API Documentation," accessed 6 June 2020. [Online]. Available: https://static.googleusercontent.com/media/www.google.com/en//covid19/exposurenotifications/pdfs/Android-Exposure-Notification-API-documentation-v1.3.2.pdf

[15] D. Leith and S. Farrell, "GAEN Due Diligence: Verifying The Google/Apple Covid Exposure Notification API," 16 June 2020. [Online]. Available: https://www.scss.tcd.ie/Doug.Leith/pubs/gaen_verification.pdf

[16] "Android Work," Accessed 11 July 2020. [Online]. Available: https://developers.google.com/android/work/

[17] "CovidTracker Data Protection Impact Assessment," 26 June 2020. [Online]. Available: https://github.com/HSEIreland/covidtracker-documentation/blob/master/documentation/privacy/DataProtectionImpactAssessmentfortheCOVIDTrackerApp-26.06.2020.pdf

[18] "Google Privacy Policy," Accessed 9 July 2020. [Online]. Available: https://policies.google.com/privacy

[19] "Firebase Help: Automatically collected user properties," Accessed 26 April 2020. [Online]. Available: https://support.google.com/firebase/answer/6317486

[20] A. Crocker, K. Opsahl, and B. Cyphers, "The Challenge of Proximity Apps For COVID-19 Contact Tracing, Electronic Frontier Foundtion," 10 April 2020. [Online]. Available: https://www.eff.org/deeplinks/2020/04/challenge-proximity-apps-covid-19-contact-tracing

[21] R. Anderson, "Contact Tracing in the Real World," 12 April 2020. [Online]. Available: https://www.lightbluetouchpaper.org/2020/04/12/contact-tracing-in-the-real-world/

[22] D. Leith and S. Farrell, "Coronavirus Contact Tracing: Evaluating The Potential Of Using Bluetooth Received Signal Strength For Proximity Detection," 6 May 2020. [Online]. Available: https://arxiv.org/abs/2006.06822

[23] ——, "Measurement-Based Evaluation Of Google/Apple Exposure Notification API For Proximity Detection In A Commuter Bus," 15 June 2020. [Online]. Available: https://www.scss.tcd.ie/Doug.Leith/pubs/bus.pdf

[24] ——, "Measurement-Based Evaluation Of Google/Apple Exposure Notification API For Proximity Detection In A Light-Rail Tram," 26 June 2020. [Online]. Available: https://www.scss.tcd.ie/Doug.Leith/pubs/luas.pdf

[25] "BLOKADA Firewall," Accessed 11 July 2020. [Online]. Available: https://blokada.org/

[26] "F-Droid," Accessed 11 July 2020. [Online]. Available: https://f-droid.org/

[27] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[28] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[29] G. P. and P. K, "On the Anonymity of Home/Work Location Pairs," in *Pervasive Computing*, 2009.

[30] M. Srivatsa and M. Hicks, "Deanonymizing mobility traces: Using social network as a side-channel," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 628–637.

[31] A. Cortesi, M. Hils, T. Kriechbaumer, and contributors, "mitmproxy: A free and open source interactive HTTPS proxy (v5.01)," 2020. [Online]. Available: https://mitmproxy.org/

[32] "Frida: Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers," Accessed 26 April 2020. [Online]. Available: https://frida.re/

[33] "Privacy and Security in Firebase," 27 Nov 2019. [Online]. Available: https://firebase.google.com/support/privacy

[34] "Google Ad Manager Help: About mobile advertising IDs," Accessed 26 April 2020. [Online]. Available: https://support.google.com/admanager/answer/6274238

[35] "Android Reference Guide: Android Id," Accessed 26 April 2020. [Online]. Available: https://developer.android.com/reference/android/provider/Settings.Secure.html#ANDROID_ID

[36] "Firebase Reference Guide: FirebaseInstanceId," Accessed 26 April 2020. [Online]. Available: https://firebase.google.com/docs/reference/android/com/google/firebase/iid/FirebaseInstanceId

[37] D. Margolis, M. Risher, B. Ramakrishnan, A. Brotman, and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)," RFC 8461 (Proposed Standard), RFC Editor, Fremont, CA, USA, Sep. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8461.txt

[38] P. Hoffman and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA," RFC 6698 (Proposed Standard), RFC Editor, Fremont, CA, USA, Aug. 2012, updated by RFCs 7218, 7671, 8749. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6698.txt

[39] Z. Durumeric, D. Adrian, A. Mirian, J. Kasten, E. Bursztein, N. Lidzborski, K. Thomas, V. Eranti, M. Bailey, and J. A. Halderman, "Neither snow nor rain nor mitm... an empirical analysis of email delivery security," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 27–39.

APPENDIX A: APP ONBOARDING FLOW

We observed the following onboarding flows for each of the apps studied (the onboarding flow may, of course, change with future app updates).

A. CoronaWarnApp

Initial "Let's Get Started" screen, then "Data Privacy" screen, then "Activate Exposure Logging" button, then "If you are diagnosed" screen, "Receive warnings" screen and finally arrive at main screen. Main screen has a "Risk Status" button, a "Notify and help" button and a "FAQ" button. It also has a menu with Overview, App Information and Settings options.

B. SwissCovid

Initial "Staying one step ahead of the virus" screen, then "Protection of privacy" screen, then "Recognising encounters using bluetooth", then "Report of potential infection", then "Tips on use" screen with a link to a data protection statement, a warning that the app does not protect against covid-19 infection and an "Accept" button. Next screen is "Allow battery optimisation", then "Activate proximity tracing", then an "The app is ready to go" screen with a "Start" button. Pressing this button leads to the main screen. The main screen has an "Encounters" button, a "Reports" button, a "What to do if .. symptoms of the disease" button and a "What to do if ... you test positive" button. The "Encounters" button leads to a screen with a toggle to enable/disable proximity tracing and a "FAQS" button, the "Reports" button leads to an information screen with a a "FAQS" button, and similarly the "What to do if .. symptoms of the disease". The "What to do if ... you test positive" button leads to a screen with an "Enter the covidcode" button.

C. Immuni

Initial “Hi!” screen, then an “Immuni takes care of you” screen, then a “Slowing down the epidemic together” screen, then a “Your privacy is protected” screen with a “Let’s get started” button. Pressing button moves to a “Your privacy is safe” screen and then to a screen with two tick boxes, one “I declare that I am at least 14 years old” and one “I have read the privacy notice”. Ticking these and clicking the “Next” button transitions to ‘ ”Which region do you live in?” screen with a list of Italian regions (one needs to be selected before the “Next” button can be clicked), then to ‘ “Which province do you live in” screen (again, one needs to be selected in order to proceed) and then to a “Enable COVID-19 exposure notifications” screen with an “Enable” button. Clicking that takes one to a “Protect your device screen” with a “Got it” button, then to a “Watch out for scam messages screen” with a “Got it” button and finally to the main screen. The main screen has several “Find out more” buttons, a “Disable the service” button and “Home” and “Settings” buttons. The home button opens the main screen, the settings button opens a screen with the options to “Report a positive result” and with “FAQ”, “Terms of use”,“Privacy notice”, “Change your province“, “Leave a review” and “Contact support” buttons.

D. StoppCorona

Initial “Welcome” screen, then a “Digital Handshake” screen, then a “Check Symptoms” screen, then a “Notification in the event of illness” screen, then a “Thank you for using the app!” screen. The next screen is “Declaration of consent” and has an “I consent” tick box and a “Finished button”. Ticking the box and clicking the button leads to a “What’s new?” page with an “All right!” button. Clicking the button turns on exposure notifications and leads to the main screen. The main screen has a toggle to enable/disable “Automatic handshake”, a “Share the app!” button, a “Check symptoms” button and a “Report medical confirmation” button. It also has a drop down menu with “Examine symptoms”, “Report medical confirmation”, “Share app”, “What can this app do”, “FAQ”, “More about coronavirus”, “Saved IDs”, “Open source licenses”, “Data privacy” and “Imprint” buttons.

E. RadarCOVID

RadarCovid is Spanish language only. The initial welcome screen has a “Continuar” button that leads to a consent page with an acceptance box that must be ticked in order to proceed. The next page has an activate bluetooth button, and pressing that leads to the main screen. The main screen has a toggle switch to enable/disable exposure notifications, a button to be pressed when tested positive with Covid-19 and an information button. Icon buttons at the bottom of the screen link to data privacy information and a contact phone number.

F. SmitteStop

SmitteStop is Danish language only. The initial screen displays a “start” button, leading to a sequence of screens with “Naeste” buttons. The last screen has a toggle switch to give consent “Godkend samtykke” that needs to be enabled in order to proceed. The next screen displays a popup asking to enable exposure notifications, and clicking ok leads to the main screen. The main screen has a button to enable/disable exposure notifications, a button to display notifications and a button to report a positive test. There is also a drop down menu with various information.

G. ApturiCovid

Initial “ApturiCovid” screen with the option to select Latvian, English and Russian and a tick box. Selecting English, the tick box is “I accept App terms of use and have read the Privacy policy”, with links to terms of use and privacy policy. Ticking this box enables the “Next” button which leads to a “Contact detection” screen with a “Switch on” toggle that says it can be switched on later. Leaving it off and clicking the “Next” button generates a popup asking to “Continue without switching on contact detection”. Clicking the toggle on changes the display to allow a phone number to be entered and to display a “I will not provide a number” button. Clicking that generates a warning popup, clicking “Yes” to that then leads to the main screen. The main screen has a toggle to enable/disable “Contact detection”, a “Share app” button, a “Home” button and three icon buttons that display infection statistics for Latvia, open a FAQ screen and open a settings screen with the options to “Enter SPKC code”, provide a phone number and a toggle to enable/disable “Notify me if contact detection gets switched off”.

H. ProteGO Safe

ProteGO Safe is Polish language only. The initial screen displays a popup with an “Ok” button, clicking this closes it. The initial screen leads through five information screens and then to a screen with a tick box that looks like a consent request. Ticking this box and proceeding leads to a page with a text entry box and two buttons, clicking on leads to a page with two buttons and then to the main page. Clicking the button at the top of the main page enables exposure notifications. The main page appears to have a button to be clicked on receiving a positive infection result, a “Home” button, two information/FAQ buttons and a settings button.

I. CovidTracker

Initial “CovidTracker” screen has an “I am 16 or older” button and a “I am under 16” button. Clicking the ‘I am 16 or older” button leads to a screen with information, a link to terms and conditions and a “Get started” button. Clicking “Get started” leads to a “Your data” screen, scrolling to the bottom shows a “Continue” button. Clicking this leads to an “App Metrics” page with “Yes, I consent” and “No thanks” buttons. Clicking “No thanks” leads to a “Contact Tracing” page where clicking “Continue” enables exposure notifications and then leads to a “Contact Tracing follow-up call” screen that allows a phone number to be entered and has “Yes, I want to opt-in” and “No thanks” buttons. Clicking “No thanks” leads to the main page. The main page has a “COVID Check-In” button, a “The

national picture" button and "Updates", "COVID Check-In", "Contact Tracing" and "Share app" buttons along the bottom and a "Settings" button at the top. The "Updates" leads to the main screen, the "COVID Check-In" button to a symptom checking page with a "Yes, I'd like to use COVID Check-In" button, the "Contact Tracing" button leads to a page with "Share app", "Close contact information" and "Upload your Random IDs" buttons. The "Settings" button opens a page with "Contact Tracing", "COVID Check-In", "Data Protection Information Notice", "Term & Conditions", "App Metrics" and "Leave" buttons.

Appendix B: Content of Network Connections

Note that to save space HTTP headers and parameters with uninteresting content are not shown. Probable persistent identifiers are highlighted in bold.

I. CoronaWarnApp (Germany) Connections

A. On First StartUp

GET https://svc90.main.px.t-online.de/version/v1/configuration/country/DE/app_config

This call is made on first startup of the app, without further user interaction. Response is application/octet-stream containing a protobuf. Once decoded it contains:

```
minRiskScore: 11
riskScoreClasses {
  risk_classes {label: "LOW" max: 15 url: "https://www.coronawarn.app"}
  risk_classes {label: "HIGH" min: 15 max: 72 url: "https://www.
coronawarn.app"}
}
exposureConfig {
  transmission {
    appDefined_1: LOWEST appDefined_2: LOW
    appDefined_3: LOW_MEDIUM appDefined_4: MEDIUM
    appDefined_5: MEDIUM_HIGH appDefined_6: HIGH
    appDefined_7: VERY_HIGH appDefined_8: HIGHEST
  }
  transmissionWeight: 50.0
  duration {
    gt_10_le_15_min: LOWEST gt_15_le_20_min: LOWEST
gt_20_le_25_min: LOWEST
    gt_25_le_30_min: LOWEST gt_30_min: LOWEST
  }
  durationWeight: 50.0
  daysSinceLastExposure {
    ge_14_days: MEDIUM_HIGH ge_12_lt_14_days: MEDIUM_HIGH
    ge_10_lt_12_days: MEDIUM_HIGH ge_8_lt_10_days:
MEDIUM_HIGH
    ge_6_lt_8_days: MEDIUM_HIGH ge_4_lt_6_days: MEDIUM_HIGH
    ge_2_lt_4_days: MEDIUM_HIGHge_0_lt_2_days: MEDIUM_HIGH
  }
  daysWeight: 20.0
  attenuation {
    gt_63_le_73_dbm: LOWEST gt_51_le_63_dbm: LOWEST
    gt_33_le_51_dbm: LOWEST gt_27_le_33_dbm: LOWEST
    gt_15_le_27_dbm: LOWEST gt_10_le_15_dbm: LOWEST
    lt_10_dbm: LOWEST
  }
  attenuationWeight: 50.0
}
attenuationDuration {
  thresholds {lower: 55 upper: 63 }
  weights{ low: 1.0 mid: 0.5 }
  riskScoreNormalizationDivisor: 25
}
appVersion {
  ios {latest { minor: 8 patch: }
    min {minor: 5}}
  android {latest {minor: 8 patch: 3}
    min {minor: 8} }
}
```

GET https://svc90.main.px.t-online.de/version/v1/diagnosis-keys/country/DE/date

Response is ["2020-06-23","2020-06-24","2020-06-25","2020-06-26","2020-06-27","2020-06-28","2020-06-29","2020-06-30","2020-07-01","2020-07-02"]

GET https://svc90.main.px.t-online.de/version/v1/diagnosis-keys/country/DE/date/2020-06-23

Response is zipped TEK files. This request is followed by a sequence of similar requests, one for each of the dates given.

B. Connections When Idle

GET https://svc90.main.px.t-online.de/version/v1/configuration/country/DE/app_config

This connectiion is typically made when the app is brought to the foreground. Response is 304 Not Modified

About once a day there is a call to check for new TEKs:

GET https://svc90.main.px.t-online.de/version/v1/diagnosis-keys/country/DE/date

Response is a list of dates, and TEKs for new dates are fetched using a GET to https://svc90.main.px.t-online.de/version/v1/diagnosis-keys/country/DE/date/<date>

II. SwissCovid (Switzerland) Connections

A. On First StartUp

GET https://www.pt.bfs.admin.ch/v1/config
Parameters:
appversion: android−1.0.5
osversion: android28
buildnr: 1592520999206

Response is json:


```
{
    "androidGaenSdkConfig": {
        "factorHigh": 0.5,
        "factorLow": 1.0,
        "higherThreshold": 55,
        "lowerThreshold": 50,
        "triggerThreshold": 15
    },
    "forceTraceShutdown": false,
    "forceUpdate": false,
    "iOSGaenSdkConfig": {
        "factorHigh": 0.5,
        "factorLow": 1.0,
        "higherThreshold": 55,
        "lowerThreshold": 50,
        "triggerThreshold": 15
    },
    "infoBox": null,
    "iosgaenSdkConfig": {
        "factorHigh": 0.5,
        "factorLow": 1.0,
        "higherThreshold": 55,
        "lowerThreshold": 50,
        "triggerThreshold": 15
    },
    "sdkConfig": {
        "badAttenuationThreshold": 73.0,
        "contactAttenuationThreshold": 73.0,
        "eventThreshold": 0.8,
        "numberOfWindowsForExposure": 3
    }
}
```

GET https://www.pt.bfs.admin.ch/v1/gaen/exposed/1593561600000

Response is zipped TEK files. This request is followed by a series of similar requests of the form GET https://www.pt.bfs.admin.ch/v1/gaen/exposed/<time>. Here <time> is the time, in seconds since 1st jan 1970, at midnight GMT and each request is for a different day.

B. Connections When Idle

GET https://www.pt.bfs.admin.ch/v1/config
Parameters:
appversion: android−1.0.5
osversion: android28
buildnr: 1592520999206

Response is json, as above.

A pair of calls checks for new TEKs:

GET https://www.pt.bfs.admin.ch/v1/gaen/exposed/¡time¿

GET https://13.224.68.88/v1/gaen/exposed/1593561600000
Parameter:
publishedafter: 1593633600000

Response is zipped TEK files

III. Immuni (Italy) Connections

A. On First StartUp

GET https://get.immuni.gov.it/v1/settings
Parameters:
platform: android
build: 1203309

Response is json:

{ ”dummy_analytics_waiting_time”: 2592000,
”dummy_teks_average_opportunity_waiting_time”: 5184000,
”dummy_teks_average_request_waiting_time”: 10,
”dummy_teks_average_start_waiting_time”: 15,
”dummy_teks_request_probabilities”: [0.95, 0.1],
”dummy_teks_window_duration”: 1209600,
”experimental_phase”: false,
”exposure_configuration”: {
”attenuation_bucket_scores”: [0,5,5,5 5,5,5,5],
”attenuation_thresholds”: [50,70],
”attenuation_weight”: 1,
”days_since_last_exposure_bucket_scores”: 1,1,1,1,1,1,1,1],
”days_since_last_exposure_weight”: 1,
”duration_bucket_scores”: [0,0,0,0,5,5,5,5],
”duration_weight”: 1,
”minimum_risk_score”: 1,
”transmission_risk_bucket_scores”: [1,1,1,1,1,1,1,1],
”transmission_risk_weight”: 1},
”exposure_detection_period”: 14400,
”exposure_info_minimum_risk_score”: 20,
”faq_url”: {”de”: ”https://get.immuni.gov.it/docs/faq−de.json”,
”en”: ”https://get.immuni.gov.it/docs/faq−en.json”,
”es”: ”https://get.immuni.gov.it/docs/faq−es.json”,
”fr”: ”https://get.immuni.gov.it/docs/faq−fr.json”,
”it”: ”https://get.immuni.gov.it/docs/faq−it.json” },
”maximum_exposure_detection_waiting_time”: 86400,
”minimum_build_version”: 1,
”onboarding_not_completed_notification_period”: 86400,
”operational_info_with_exposure_sampling_rate”: 1,
”operational_info_without_exposure_sampling_rate”: 0.6,
”pn_url”: { ”de”: ”https://www.immuni.italia.it/app−pn.html”,
”en”: ”https://www.immuni.italia.it/app−pn.html”,
”es”: ”https://www.immuni.italia.it/app−pn.html”,
”fr”: ”https://www.immuni.italia.it/app−pn.html”,
”it”: ”https://www.immuni.italia.it/app−pn.html”},
”required_update_notification_period”: 86400,
”risk_reminder_notification_period”: 86400,
”service_not_active_notification_period”: 86400,
”support_email”: ”cittadini@immuni.italia.it”,
”support_phone”: ”800912491”,
”support_phone_closing_time”: ”22”,
”support_phone_opening_time”: ”7”,
”teks_max_info_count”: 600,
”teks_max_summary_count”: 84,
”teks_packet_size”: 110000,
”tou_url”: { ”de”: ”https://www.immuni.italia.it/app−tou.html”,
”en”: ”https://www.immuni.italia.it/app−tou.html”,
”es”: ”https://www.immuni.italia.it/app−tou.html”,
”fr”: ”https://www.immuni.italia.it/app−tou.html”,
”it”: ”https://www.immuni.italia.it/app−tou.html”}
}

GET https://get.immuni.gov.it/docs/faq-en.json

Response is json containing FAQ text.

GET https://get.immuni.gov.it/v1/keys/index

Response is json:

{ ”newest”: 19,
”oldest”: 9}

GET https://get.immuni.gov.it/v1/keys/9

Response is zipped TEKs. This is followed by a sequence of similar GET requests to get.immuni.gov.it/v1/keys/<number> where counts from the oldest to newest value in previous response.

B. Connections When Idle

Immuni does not generate new connections upon being brought to the foreground. Instead, roughly every 6 hours the app repeats the following calls:

GET https://get.immuni.gov.it/v1/settings
Parameters:
platform: android
build: 1203309

GET https://get.immuni.gov.it/docs/faq-en.json

GET https://get.immuni.gov.it/v1/keys/index

And fetches any new zipped TEK files using GET requests to get.immuni.gov.it/v1/keys/<number>

IV. StoppCorona (Austria) Connections

A. On First StartUp

GET https://app.prod-rca-coronaapp-fd.net/Rest/v8/configuration
Headers:
authorizationkey: 64165cfc5a984...
x−appid: at.roteskreuz.stopcorona

The authorizationkey header value is hard-wired in the app and so is the same for all instances of the app. The response is json which includes question text and the following configuration settings:

”exposure_configuration”: {
”attenuation_duration_thresholds”: [33,63],
”attenuation_level_values”: [0,1,2,2,8,8,8,8],
”daily_risk_threshold”: 7,
”days_since_last_exposure_level_values”: [1,1,1,1,1,1,1,1],
”duration_level_values”: [0,1,2,3,4,5,6,7],
”minimum_risk_score”: 1,
”transmission_risk_level_values”: [1,1,1,1,1,1,1,1]
},

GET https://cdn.prod-rca-coronaapp-fd.net/exposures/at/index.json
Headers:
authorizationkey: 64165cfc5a984...
x−appid: at.roteskreuz.stopcorona

Response is json:

{”full_14_batch”:{”interval”:2654928,”batch_file_paths”:[”/exposures/at/1594235700/batch_full14−2654928−1.zip”]},”full_7_batch”:{”interval”:2655936,”batch_file_paths”:[”/exposures/at/1594235700/batch_full7−2655936−1.zip”]},”daily_batches”:[{”interval”:2655936,”batch_file_paths”:[”/exposures/at/1594235700/batch−2655936−1.zip”]},{”interval”:2656080,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656080−1.zip”]},{”interval”:2656224,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656224−1.zip”]},{”interval”:2656368,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656368−1.zip”]},{”interval”:2656512,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656512−1.zip”]},{”interval”:2656656,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656656−1.zip”]},{”interval”:2656800,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656800−1.zip”]},{”interval”:2656944,”batch_file_paths”:[”/exposures/at/1594235700/batch−2656944−1.zip”]}]}

GET https://cdn.prod-rca-coronaapp-fd.net/exposures/at/1594235700/batch_full14-2654928-1.zip
Headers:
authorizationkey: 64165cfc5a984bb...
x−appid: at.roteskreuz.stopcorona

Response is a zipped TEK file.

B. Connections When Idle

A few times a day the app makes requests to check for new published TEKs:

GET https://cdn.prod-rca-coronaapp-fd.net/exposures/at/index.json
Headers:
authorizationkey: 64165cfc5a984bb...
x−appid: at.roteskreuz.stopcorona

GET https://cdn.prod-rca-coronaapp-fd.netexposures/at/1594275300/batch_full14-2655072-1.zip
Headers:
authorizationkey: 64165cfc5a984bb...
x−appid: at.roteskreuz.stopcorona

V. RadarCOVID (Spain) Connections

A. On First StartUp

On first startup the app fetches configuration settings:

GET https://dqarr2dc0prei.cloudfront.net/configuration/settings

The response is json:

{ ”applicationVersion”: {
”android”: {
”bundleUrl”: ”https://play.google.com/store/apps/details?id=es.gob.radarcovid”,
”compilation”: 3,
”version”: ”1.0.0”},
”ios”: {”bundleUrl”: ”itms://itunes.apple.com/app/id1520443509”,
”compilation”: 0,
”version”: ”1.0.0”}},
”attenuationDurationThresholds”: { ”low”: 53,”medium”: 60},
”attenuationFactor”: {”low”: 1.0,”medium”: 0.5 },
”exposureConfiguration”: {
”attenuation”: {
”riskLevelValue1”: 1,”riskLevelValue2”: 1,
”riskLevelValue3”: 1,”riskLevelValue4”: 1,
”riskLevelValue5”: 1,”riskLevelValue6”: 1,
”riskLevelValue7”: 1,”riskLevelValue8”: 1,
”riskLevelWeight”: 100.0,
”days”: {
”riskLevelValue1”: 0,”riskLevelValue2”: 0,
”riskLevelValue3”: 0,”riskLevelValue4”: 0,
”riskLevelValue5”: 0,”riskLevelValue6”: 0,
”riskLevelValue7”: 0,”riskLevelValue8”: 0,
”riskLevelWeight”: 0.0},
”duration”: {
”riskLevelValue1”: 0,”riskLevelValue2”: 0,
”riskLevelValue3”: 0,”riskLevelValue4”: 0,
”riskLevelValue5”: 0,”riskLevelValue6”: 0,
”riskLevelValue7”: 0,”riskLevelValue8”: 0,
”riskLevelWeight”: 0.0},
”transmission”: {
”riskLevelValue1”: 0,”riskLevelValue2”: 0,
”riskLevelValue3”: 0,”riskLevelValue4”: 0,
”riskLevelValue5”: 0,”riskLevelValue6”: 0,
”riskLevelValue7”: 0,”riskLevelValue8”: 0,
”riskLevelWeight”: 0.0}},
”minDurationForExposure”: 15,
”minRiskScore”: 1,
”riskScoreClassification”: [
{”label”: ”LOW”,”maxValue”: 4095,”minValue”: 0},
{”label”: ”HIGH”,”maxValue”: 4096,”minValue”: 4096 }
]
}

GET https://dqarr2dc0prei.cloudfront.net/configuration/token/uuid

The response is json:

{”uuid”: ”9a18360e−bcb2−4bbe−a687−d26c98d3d173”}

We do not observe this uuid being sent back to the server in later requests so its purpose remains unclear. The app now starts checking for published keys for infected people:

GET
https://dqarr2dc0prei.cloudfront.net/dp3t/v1/gaen/exposed/1594512000000
Header:
user−agent: dp3t−sdk−android

Note that 1594512000000 is the time in seconds since 1 Jan 1970 that corresponds to 12 July 2020 0.00 GMT, the day on which the measurements were taken. The request URL therefore appears to have a similar format to that used by SwissCovid. The response is 204 No Content with headers:

x−public−key: LS0tLS1CRUdJTiBQV...
signature: eyJhbGciOiJFUzI1NiJ9.ey...

However, inspection of the decompiled app code suggests that these header values are not used by the app. The app then makes a sequence of similar requests, changing the query time to midnight on 11 July, 10 July and so on back to 3 July. These return zipped TEK files that seem to contain dummy TEKs (they are signed with org.dpppt.ios.demo)

B. Connections When Idle

The app makes the same calls to download configuration settings and check for publication of new keys when idle.

VI. SmitteStop (Denmark) Connections

A. On First StartUp

On first startup we observed no network connections being made by SmitteStop.

B. Connections When Idle

The app makes the following three calls a few times a day. The first request fetches updates to the published diagnoses keys:

GET https://app.smittestop.dk/API/v1/diagnostickeys/2020-07-12:0.zip
Headers:
Authorization_Mobile: 68iXQyx...
Manufacturer: Google
OSVersion: 9
OS: Android−Google

The “Authorization_Mobile” header value is hard-wired in the app and so is the same for all instances. The response is a zipped file containing TEKs. The second request fetches configuration settings:

GET https://app.smittestop.dk/API/v1/diagnostickeys/exposureconfiguration
Headers:
Authorization_Mobile: 68iXQyx...
Manufacturer: Google
OSVersion: 9
OS: Android−Google
Cookie: NSC_ejh−ttbq−mcb02−443=
ffffffff0909d48745525d5f4f58455e445a4a423660

Note the cookie sent with the request. The response sets/refreshes the cookie:
Set−Cookie: NSC_ejh−ttbq−mcb02−443=
ffffffff0909d48745525d5f4f58455e445a4a423660;expires=Mon, 13−Jul
−2020 11:53:08 GMT;path=/;secure;httponly

TThe cookie value appears, however, to be the same for all instances of the app. The response body is json:

{ ”attenuationScores”: [1,1,1,8,8,8,8,8],
”attenuationWeight”: 50,
”daysSinceLastExposureScores”: [1,1,1,1,1,1,1,1],
”daysSinceLastExposureWeight”: 50,
”durationAtAttenuationThresholds”: [51,63],
”durationScores”: [1,1,1,1,8,8,8,8],
”durationWeight”: 50,
”minimumRiskScore”: 512,
”transmissionRiskScores”: [1,1,8,8,8,8,1,1],
”transmissionRiskWeight”: 50}

The third request sends log event data:

POST https://app.smittestop.dk/API/v1/logging/logMessages
Headers:
Authorization_Mobile: 68iXQyx...
Manufacturer: Google
OSVersion: 9
OS: Android−Google
The request body various, but contains various logging messages, for example:

{
”logs”: [
{
”additionalInfo”: ””,
”api”: null,
”apiErrorCode”: null,
”apiErrorMessage”: null,
”apiVersion”: 1,
”buildNumber”: ”87”,
”buildVersion”: ”1.0.2”,
”description”: ”Failed at adding a status to the latest tuple in SaveStatusOfLastPullCall”,
”deviceCorrelationId”: ””,
”deviceDescription”: ”Pixel 2”,
”deviceOSVersion”: ”9”,
”deviceType”: ”Android−Google”,
”exceptionMessage”: ”Value cannot be null. Parameter name: value”,
”exceptionStackTrace”: ”at Newtonsoft.Json.JsonConvert.DeserializeObject (System.String value, System.Type type, Newtonsoft.Json.JsonSerializerSettings settings) <0x77a5978f9c + 0x+xx8> in <207xxxxxxxxx917a5e8f91b0b239405>:0 at Newtonsoft.Json.JsonConvert.DeserializeObject[T] (System.String value, Newtonsoft.Json.JsonSerializerSettings settings) <0x77a5978e50 + 0x+xx3> in <207xxxxxxxxx917a5e8f91b0b239405>:0 at Newtonsoft.Json.JsonConvert.DeserializeObject[T] (System.String value) <0x77a5978cb8 + 0x+xxb> in <207xxxxxxxxx917a5e8f91b0b239405>:0 at NDB.Covid19.Base.AppleGoogle.ExposureNotification.Helpers.FetchExposureKeys.StoredValueHelper.SaveStatusOfLastPullCall (System.String status) <0x77a4e31a34 + 0x+xx7> in <2287135a28b9489c8c80db20eeff6feb>:0
”,
”exceptionType”: ”ArgumentNullException”,
”innerExceptionMessage”: null,
”innerExceptionStackTrace”: null,
”innerExceptionType”: null,
”reportedTime”: ”2020−07−12T12:53:07.951068+01:00”,
”severity”: ”ERROR”
}
]
}

VII. ApturiCovid (Latvia) Connections

A. On First StartUp

The app initialises Firebase upon first startup, which generates the following POST request (standard/uninteresting parameters/headers are omitted):

POST https:
//firebaseinstallations.googleapis.com/v1/projects/apturicovid/installations
Headers:
X−Android−Package: lv.spkc.gov.apturicovid
Request body:
”fid”: ”**c8WD_GfwSvWv89aGYrWd9d**”,

The “fid” value is the Firebase Instance ID, which uniquely identifies the current instance of the app. The response to this request echoes the fid value and includes two tokens which appear to be used to identify the current session.

Response is json:

{
”authToken”: {
”expiresIn”: ”604800s”,
”token”: ”**eyJhbGciOiJ...Uw**”
},
”fid”: ”**c8WD_GfwSvWv89aGYrWd9d**”,
”name”: ”projects/503076402814/installations/**c8WD_GfwSvWv89aGYrWd9d**”,
”refreshToken”: ”2_FSe0x3z9Dv...”
}

The fid, the token and what seems to be a device identifier are now sent to Google:
POST https://android.clients.google.com/c2dm/register3
Headers:
Authorization: AidLogin **4501126624510942234**:4321116450608895094
app: lv.spkc.gov.apturicovid
X−appid: **c8WD_GfwSvWv89aGYrWd9d**
X−Goog−Firebase−Installations−Auth:**eyJhbGciOiJFUzI1NiIsI...Uw**
app: lv.spkc.gov.apturicovid
device: **4501126624510942234**
info: Y4Gga0ZwyogVQEb1aBJ6XhyidEyqqhY

The response is a new token that includes the fid:
token=**c8WD_GfwSvWv89aGYrWd9d**:APA91bGC...

The first call to app-measurement.com is now made:
GET https://app-measurement.com/config/app/1\%3A503076402814\
%3Aandroid\%3A5d8cba16d971f6221aca2f
Parameters:
app_instance_id: **f5fd7266cb7df65bdf87d6788d550c3e**

Response is json.

And now telemetry data is sent to app-measurement.com. The payload includes the Google advertising id of the device (1d2635f5-2af7-4fb3-86e8-5fd6...), the app_instance_id and the fid:

POST https://app-measurement.com/a
<...>
\x02_o\x12\x04auto
\x17
\x04_cis\x12\x0freferrer API v2\x12\x04_cmp\x18\x96\x98\xe6\xd9\xb1. \x00\x1a\x14\x08\xcd\xde\xe5\xd9\xb1.\x12\x04_fot \x80\x8c\xa8\xdb\xb1.\x1a\x0e\x08\xcd\xde\xe5\xd9\xb1.\x12\x03_fi \x01\x1a\x0f\x08\xac\xdf\xe5\xd9\xb1.\x12\x04_sno \x01\x1a\x13\x08\xac\xdf\xe5\xd9\xb1.\x12\x04_sid \xcf\xa6\x83\xf8\x05\x1a\x0f\x08\xb8\xe8\xe6\xd9\xb1.\x12\x04_lte \x01\x1a\x0e\x08\xb8\xe8\xe6\xd9\xb1.\x12\x03_se \x00 \xa4\xe8\xe6\xd9\xb1.(\xac\xdf\xe5\xd9\xb1.0\x96\x98\xe6\xd9\xb1.8\xcd\xde\xe5\xd9\xb1.B\x07androidJ\x019R\x07Pixel 2Z\x05en−us‘<r\x17lv.spkc.gov.apturicovid\x82\x01\x061.0.47\x88\x01\xe0\xda\x01\x90\x01\x85\xab\x0c\x9a\x01$**1d2635f5-2af7-4fb3-86e8-5fd6...**\xa0\x01\x00\xaa\x01**f5fd7266cb7df65bdf87d6788d550c3e**\xb0\x01\x84\x87\xdc\xaa\xf9\xa3

\xeb\xf6\xaa\x01\xb8\x01\x02\xca\x01−1:503076402814:android:5
d8cba16d971f6221aca2f\xd0\x01\xcd\xde\xe5\xd9\xb1.\xe0\x01\x01\
xf2\x01\x16c**8WD_GfwSvWv89aGYrWd9d**\xf8\x01/\x98\x02\xec\xae
\x99\x93\x80\x84\xea\x02\xe8\x02\xc8\xeb\x86\x0b\xf0\x02

Finally, the app gets around to downloading exposure notification data. The first call checks for published TEKs, there are none so the response is empty:

GET https://apturicovid-files.spkc.gov.lv/dkfs/v1/index.txt
Response is 200 OK with empty content.

Now the exposure notification configuration settings are fetched:
GET
https://apturicovid-files.spkc.gov.lv/exposure_configurations/v1/android.json

Response is json:

{ ”attenuation_scores”: [1,2,3,4,5,6,7,8],
”attenuation_threshold”: [50,55],
”attenuation_weight”: 1,
”days_since_last_exposure_scores”: [1,1,1,1,1,1,1,1],
”days_since_last_exposure_weight”: 1,
”duration_scores”: [1,1,1,1,8,8,8,8],
”duration_weight”: 1,
”minimum_risk_score”: 30,
”transmission_risk_scores”: [1,1,1,1,1,1,1,1],
”transmission_risk_weight”: 1 }

GET https://apturicovid-files.spkc.gov.lv/stats/v1/covid-stats.json
Response is json:

{ ”infected_tests_proportion”: 0.72,
”total_death_count”: 30, ”total_infected_count”: 1123,
”total_tests_count”: 156784,”updated_at”: ”2020−07−04”,
”yesterday_death_count”: 0,”yesterday_infected_count”: 1,
”yesterday_tests_count”: 1185}

B. Connections When Idle

The app makes the following calls several times a day. Firstly, three calls to fetch config and exposure information:

GET https://apturicovid-files.spkc.gov.lv/dkfs/v1/index.txt
Response is 304 Not Modified.

GET
https://apturicovid-files.spkc.gov.lv/exposure_configurations/v1/android.json
Response is 304 Not Modified.

GET https://apturicovid-files.spkc.gov.lv/stats/v1/covid-stats.json
Response is 304 Not Modified.

Then two calls to send telemetry data to app-measurement.com. The first appears to be checking for updates, to which the response is 304 Not Modified :

GET https://app-measurement.com/config/app/1\%3A503076402814\
%3Aandroid\%3A5d8cba16d971f6221aca2f
Parameters:
app_instance_id: **f5fd7266cb7df65bdf87d6788d550c3e**

The second connection send telemetry data logging user interactions with the app. The data is sent using a standard Firebase format and includes the Google advertising id of the device (1d2635f5-2af7-4fb3-86e8-5fd6...), the app_instance_id and the fid:

POST https://app-measurement.com/a
<...>
\x03_si\x18\xdc\x98\xe8\xfd\xc3\xda\xbb\xb5\x97\x01\x12\
x02_e\x18\xb7\xcb\xdb\xdb\xb1. \xc3\xfa\xda\xdb\xb1.\x1a\x14\x08
\xcd\xde\xe5\xd9\xb1.\x12\x04_fot \x80\x8c\xa8\xdb\xb1.\x1a\x0e\
x08\xcd\xde\xe5\xd9\xb1.\x12\x03_fi \x01\x1a\x11\x08\xbf\xc2\xdc
\xdb\xb1.\x12\x04_lte \xfe\x9e9\x1a\x0f\x08\x9a\xe9\xda\xdb\xb1.\
x12\x04_sno \x02\x1a\x13\x08\x9a\xe9\xda\xdb\xb1.\x12\x04_sid \
xff\xc5\x83\xf8\x05\x1a\x0f\x08\xc0\xc2\xdc\xdb\xb1.\x12\x03_se \
xfaH \xa6\xc2\xdc\xdb\xb1.(\x9a\xe9\xda\xdb\xb1.0\xb7\xcb\xdb\
xdb\xb1.8\xe7\x81\xb5\xda\xb1.B\x07androidJ\x019R\x07Pixel 2Z\
x05en−us‘<r\x17lv.spkc.gov.apturicovid\x82\x01\x061.0.47\x88\x01\
xe0\xda\x01\x90\x01\x85\xab\x0c\x9a\x01$**1d2635f5-2af7-4fb3-86e8-5fd6e53f2aff**\xa0\x01\x00\xaa\x01
f5fd7266cb7df65bdf87d6788d550c3e\xb0\x01\x84\x87\xdc\xaa\xf9\xa3
\xeb\xf6\xaa\x01\xb8\x01\x04\xca\x01−1:503076402814:android:5
d8cba16d971f6221aca2f\xd0\x01\xc1\xd1\xe7\xd9\xb1.\xe0\x01\x01\
xf2\x01\x16c8WD_GfwSvWv89aGYrWd9d\xf8\x01/\x98\x02\xec\xae
\x99\x93\x80\x84\xea\x02\xe8\x02\xc8\xeb\x86\x0b\xf0\x02

VIII. ProteGO Safe (Poland) Connections

A. On First StartUp

The app initialises Firebase upon first startup:

POST https://firebaseinstallations.googleapis.com/v1/projects/safesafe-app/installations
Headers:
X−Android−Package: pl.gov.mc.protegosafe
Request body:
”fid”: ”**cY46N0YUR_ykuI0m_Bx-kz**”,

The “fid” value is the Firebase Instance ID, which uniquely identifies the current instance of the app. The response to this request echoes the fid value and includes two tokens.

Response is json:

{ ”authToken”: {
”expiresIn”: ”604800s”,
”token”: ”**eyJhbGciOiJFUzI1N...kY**”
},
”fid”: ”**cY46N0YUR_ykuI0m_Bx-kz**”,
”name”: ”projects/466787798978/installations/
cY46N0YUR_ykuI0m_Bx-kz”,
”refreshToken”: ”2_WCDG3OKZQD7DQJ...”}

The fid, the token and what seems to be a device identifier are now sent to Google:
POST https://android.clients.google.com/c2dm/register3
Headers:
Authorization: AidLogin **4501126624510942234**:4321116450608895094
app: lv.spkc.gov.apturicovid
X−appid: **cY46N0YUR_ykuI0m_Bx-kz**
X−Goog−Firebase−Installations−Auth:**eyJhbGciOiJFUzI1N...kY**
app: pl.gov.mc.protegosafe
device: **4501126624510942234**
info: Y4Gga0ZwyogVQEb1aBJ6XhyidEyqqhY

The response is “Error=FIS_AUTH_ERROR”.

The app now makes a series of calls to Google’s SafetyNet API. The first call sends data that includes the device hardware serial number HT85G1A05...:

POST https://www.googleapis.com/androidantiabuse/v1/x/create
Headers:
User−Agent: DroidGuard/202117028
Request body:
<...>
\x06SERIAL\x12\x0c**HT85G1A05...**\x12\x14
<...>

The response is a large (44KB) protobuf. A protobuf is now sent to Google:

POST https://www.googleapis.com/androidcheck/v1/attestations/attest
Headers:
User−Agent: SafetyNet/202117028 (walleye PPR2.180905.005); gzip

The response looks like a signature. This is not forwarded to the app backend server for verification, which would normally be the next step in use of the SafetyNet service. The app now calls:

POST https://android.clients.google.com/c2dm/register3
Headers:
app: pl.gov.mc.protegosafe
Request body:
X−appid: **cY46N0YUR_ykuI0m_Bx-kz**
X−Goog−Firebase−Installations−Auth:**eyJhbGciOiJFUzI1N...kY**

device: **4501126624510942234**

Observe that Tthe X-appid value is the fid and the device seems to be a persistent identifer associated with the handset, perhaps the androidID (which set on first startup of a device). The response is a token based on the fid:
token=**cY46N0YUR_ykuI0m_Bx-kz**:APA91...

The app now fetches exposure configuration information from Firebase:

POST https://firebaseremoteconfig.googleapis.com/v1/projects/466787798978/namespaces/firebase:fetch
Request body:
<...>
”appInstanceId”: ”**cY46N0YUR_ykuI0m_Bx-kz**”,
”appInstanceIdToken”: ”**cY46N0YUR_ykuI0m_Bx-kz:APA91...c**”,
”packageName”: ”pl.gov.mc.protegosafe”,

The response is json:
{”appName”: ”pl.gov.mc.protegosafe”,
”entries”: {
”diagnosisKeyDownloadConfiguration”: ”{\”timeoutMobileSeconds\”:120,\”timeoutWifiSeconds\”:60,\”retryCount\”:2}”,
”exposureConfiguration”: ”{\”minimumRiskScore\”:4,\”attenuationScores\”:[2,5,6,7,8,8,8,8],\”attenuationWeigh\”:50,\”daysSinceLastExposureScores\”:[7,8,8,8,8,8,8,8],\”daysSinceLastExposureWeight\”:50,\”durationScores\”:[0,5,6,7,8,8,8,8],\”durationWeight\”:50,\”transmissionRiskScores\”:[8,8,8,8,8,8,8,8],\”transmissionRiskWeight\”:50,\”durationAtAttenuationThresholds\”:[48,58]}”,
”provideDiagnosisKeysWorkerConfiguration”: ”{\”repeatIntervalInMinutes\”:360,\”backoffDelayInMinutes\”:10}”,
”riskLevelConfiguration”: ”{\”maxNoRiskScore\”:0,\”maxLowRiskScore\”:1499,\”maxMiddleRiskScore\”:2999}”
},
”state”: ”UPDATE”}

The app now fetches information on published exposure keys:
GET https://exp.safesafe.app/%2Findex.txt
The response is:
/1592740800−00001.zip
/1592784000−00001.zip
/1593129600−00001.zip
/1593172800−00001.zip

B. Connections When Idle

The app makes the following call several times a day:

POST https://firebaseremoteconfig.googleapis.com/v1/projects/466787798978/namespaces/firebase:fetch
Request body:
<...>
”appInstanceId”: ”**cY46N0YUR_ykuI0m_Bx-kz**”,
”appInstanceIdToken”: ”**cY46N0YUR_ykuI0m_Bx-kz:APA91...c**”,
”packageName”: ”pl.gov.mc.protegosafe”,

The response is json: {”state”: ”NO_CHANGE”}

The following call is made less frequently:

GET https://exp.safesafe.app/%2Findex.txt

The response is as above.

IX. CovidTracker (Ireland) Connections

A. On First StartUp

On first startup (before any user interaction) the app initialises Firebase:

POST https://firebaseinstallations.googleapis.com/v1/projects/api-7164394121131961544-290715/installations
Headers:
X−Android−Package: com.covidtracker.hse
Request body:
”fid”: ”**do6qB-2BSDSRi2XLZIr-ul**”

The “fid” value is the Firebase Instance ID, which uniquely identifies the current instance of the app. The response to this request echoes the fid value and includes two tokens.

Response is json:
{ ”authToken”: {
”expiresIn”: ”604800s”,
”token”: ”**eyJhbGciOiJFUzI1N...0Q**”
},
”fid”: ”**do6qB-2BSDSRi2XLZIr-ul**”,
”name”: ”projects/1087125483031/installations/**do6qB-2BSDSRi2XLZIr-ul**”,
”refreshToken”: ”**2_7PZMhQEHJjyKfseL...**”}

The now app now sends the fid and a persistent device identifier (likely the Android ID, which is set on first startup of a device and only changes upon a factory reset) toi android.clients.google.com :

POST https://android.clients.google.com/c2dm/register3
Headers:
app: com.covidtracker.hse
Request body:
app: com.covidtracker.hse
X−appid: **do6qB-2BSDSRi2XLZIr-ul**
device: **4501126624510942234**

The response is “Error=FIS_AUTH_ERROR”. The app now downloads exposure notification configuration data:.

GET https://app.covidtracker.ie/api/settings

The response is 56KB of json containing data protection text etc plus the following config settings:
”exposureCheckInterval”: ”120”,
”exposureConfig”: ”{\”minimumRiskScore\”:1,\”attenuationLevelValues\”:[2,3,4,5,6,7,8,8],\”attenuationWeight\”:1,\”daysSinceLastExposureLevelValues\”:[1,1,1,1,1,1,1,1],\”daysSinceLastExposureWeight\”:1,\”durationLevelValues\”:[1,1,1,1,1,1,1,1],\”durationWeight\”:1,\”transmissionRiskLevelValues\”:[1,1,1,1,1,1,1,1],\”transmissionRiskWeight\”:1, \”durationAtAttenuationThresholds\”: [56,62],\”thresholdWeightings\”:[1,1,0],\”timeThreshold\”:15}”,

When the user navigates to the consent page within the app and agrees the app then makes a sequence of requests to Google’s SafetyNet API.

POST https://app.covidtracker.ie/api/register
Request body:
{}

The next call sends data that includes the device hardware serial number HT85G1A05... to Google:

POST https://www.googleapis.com/androidantiabuse/v1/x/create
Headers:
User−Agent: DroidGuard/202117028
Request body:
<...>
\x06SERIAL\x12\x0c**HT85G1A05...**\x12\x14
<...>

The response is a large (185KB) protobuf that looks like it contains a program executable. A 29KB protobuf is then sent to Google:

POST https://www.googleapis.com/androidcheck/v1/attestations/attest
Headers:
X−Android−Package: com.covidtracker.hse
User−Agent: SafetyNet/202117028 (walleye PPR2.180905.005); gzip

The response looks like a base64 encoded payload which includes certificate details. This is then forwarded as the payload in a PUT:

PUT https://app.covidtracker.ie/api/register

The response is json containing a "refreshToken" and "token" values. The app now calls:

GET https://app.covidtracker.ie/api/stats
Headers:
authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k

The value of the "authorization: Bearer" matches the "token" value returned in the response to the previous request. This same value is sent in all subsequent requests. The response is 17KB of json containing infection statistics data. The app now checks for published keys for infected people:

GET https://app.covidtracker.ie/api/exposures/
Headers:
authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Parameters:
since: 0
limit: 1

The response is:

```
[
    {
        "id": 6,
        "path": "exposures/ie/1594296005512.zip"
    }
]
```

and then fetches the keys themselves:

GET https://app.covidtracker.ie/api/data/exposures/ie/1594296005512.zip
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k

The response is a zipped file containing TEKs. The app also sends some telemetry data:

POST https://app.covidtracker.ie/api/metrics
Headers:
authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Request body:
{ "event": "CALLBACK_OPTIN",
"os": "android",
"version": "1.0.40"}

POST https://app.covidtracker.ie/api/metrics
Headers:
authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Request body:
{ "event": "DAILY_ACTIVE_TRACE",
"os": "android",
"version": "1.0.40"}

B. Connections When Idle

The app makes the following call approximately every 2 hours, and also sometimes when the app is brought to the foreground:

GET https://app.covidtracker.ie/api/exposures/
Headers:
authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI...4k
Parameters:
since: 0
limit: 1

The response is "[]".

X. Google Play Services Connections

Google Play Services sends data to Google roughly every 10 minutes using the following request:

POST https://play.googleapis.com/p/log/batch
Headers:
Authorization: Bearer **ya29.a0AfH6SMD_tttV5NQ...UFA**
X−SERVER−TOKEN: **CAESKQDyi0h8u1NFMSbtIY...**
Cookie: NID=**204=cZfpA_V3EeVcgH8ON4DUR...**
User−Agent: Android/com.google.android.gms/202117028 (walleye PPR2.180905.005); gzip

Note the cookie, Authorization and X-SERVER-TOKEN headers sent with this request (the cookie is set by the response to a previous request to play.googleapis.com/p/log/batch), which allows requests to be easily linked together. The request body is binary encoded, often quite large, and the content changes. It appears to contain details and telemetry of apps and services running on the handset. This includes fine-grained details of errors logged, including execution backtraces, details of network connection errors etc, and information on all apps and services installed. Some examples of information sent are as follows.

Information about the Exposure Notification service within Google Play Services:

\x10\x0e\x18\x00\x12\x06\x08\x0e\x10\x0f\x18\x00\x12\x06\x08\
x0f\x10\x10\x18\x00\x12\x06\x08\x10\x10\x11\x18\x00\x12\x06\
x08\x11\x10\x12\x18\x00\x12\x06\x08\x12\x10\x13\x18\x00\x12\
x06\x08\x13\x10\x14\x18\x00\x12\x06\x08\x14\x10\x1e\x18\x00\
x12\x06\x08\x1e\x10x\x18\x00\x12\x04\x08x\x18\x00:\x00J\x02\
x08\x01R\x00Z\x17ch.admin.bag.dp3t_10050‘\x00j)
#rolling_proximity_id_key_size_bytes\x12\x0216j)
"max_interpolation_duration_seconds\x12\x03600j(
"scan_interval_random_range_seconds\x12\x0290j\x1d
\x17timestamp_noise_minutes\x12\x0230j\x0f
\x07enabled\x12\x04truej:
1ble_scheduler_use_scheduled_executor_max_delay_ms\x12\x0530000j
\x1c
\x15scan_interval_seconds\x12\x03300j−
(min_exposure_bucketized_duration_seconds\x12\x011j\x0e
scan_mode\x12\x012j3
−associated_metadata_encryption_key_size_bytes\x12\x0216j,
&default_min_exposure_attenuation_value\x12\x0247j\x17
\x12advertise_tx_power\x12\x011j\x16
\x11scan_time_seconds\x12\x014j7
2risk_score_bucketed_transmission_risk_scores_count\x12\x018j\x17
\x11contact_id_length\x12\x0216j8
0data_quality_log_ble_advertising_stats_only_once\x12\x04truej\x13
\x0eadvertise_mode\x12\x011j.
\%reschedule_start_scan_min_interval_ms\x12\x0510000j
\x18wake_up_for_ble_executor\x12\x04truej\x1e
\x15advertise_connectable\x12\x05falsej−
'id_rotation_period_random_range_seconds\x12\x0290j.
&data_quality_log_ble_advertising_stats\x12\x04truej6
/tk_rolling_period_multiple_of_id_rolling_period\x12\x03144j&
temporary_tracing_key_size_bytes\x12\x0216j$
\x1brequire_multi_advertisement\x12\x05falsej\x1f
\x19id_rolling_period_minutes\x12\x0210j
\x1aid_rotation_period_seconds\x12\x03690j&
\x1fdata_quality_log_period_minutes\x12\x03240j$
\x1eclearcut_upload_period_minutes\x12\x0230\x80\x01\x01\x9a\
x01<
\x04\x10\x01\x18\x00
\x07\x08\x01\x10\x90N\x18\x00
\x08\x90N\x10\xa0\x8d\x06\x18\x00
<...>
\x05\x08\x90N\x18\x00x\xa08\x88\x01\xb5\x8c\xde\x14\xb0\
x01−\xba\x01\x04\x08\x01\x10\x002\
x1cNEARBY_EXPOSURE_NOTIFICATION:\x00@\xf2\x97\xf7\x14H\
x02P\x05Z\x08\x08\x01\x10\x01\x18\x02 \x00j\x06\x08\x00\x10\
x00\x18\x00\x18\x01

What appears to be telemetry from the Exposure Notification service and the CovidTracker app:

9,10168,l,wl,nearby:ExposureNotificationScanner,0,f,0,0,0,0,0,p,0,0,0,0,0,
bp,0,0,0,0,0,0,w,0,0,0,0
9,10168,l,wl,∗alarm∗,0,f,0,0,0,0,0,p,0,0,0,0,0,bp,0,0,0,0,0,w,0,0,0,0

9,10168,l,wl,∗launch∗,0,f,0,0,0,0,0,p,0,0,0,0,0,0,bp,0,0,0,0,0,0,w,0,0,0,0
9,10178,l,wl,∗job∗/com.covidtracker.hse/androidx.work.impl.background.systemjob.SystemJobService,0,f,0,0,0,0,0,p,0,0,0,0,0,0,bp,0,0,0,0,0,0,w,0,0,0,0
27

Plus phone identifiers, including the SIM serial number (8935311180135555..) and the phone number (+353892197...):

\x80\xd3\x99\xfa\xb0.∗\xa5\x1e\xee\x02\x08\x9f\xd6\xbe\xf7\xb0.2\x7f\x12\x00\x1a(\x08\x88\xa0\xd5\x1c\x12 6.0.117 (Xorn_RC10.phone_dynamic)"\x02
\x00\xb2\x01I
G\x08\x01\x10\x00\x18\x02"\x0527211∗\x0c\x08\x01\x10\x02\x1a\x06\x08\x01\x10\x01\x18\x012\x13**8935311180135555...**:\x13**8935311180135555...**H\x01\xd0\x01\x03X\x0ex\xa08\x88\x01\x85\xec\xa5\x1d\x82\x01\xcd\x01\x12\xac\x01\x08\x03\x12\xa7\x
<...>
\xa3\xa4\xfa\x80\xb4.∗\xbc[\x80\x03\x08\xed\xef\xea\xfd\xb3.2\x8e\x01\x12\x00\x1a(\x08\xa8\x88\x91\x1d\x126.1.097 (Yeti_RC11.phone_dynamic)"\x0f

+353892197...\xb2\x01K

A checkin request is made roughly once per day. This request is quite worriesome since it includes several hardware identifiers including the phone IMEI (35753708924...), the hardware serial number (HT85G1A05...), the SIM serial number and the user email address (highlighted in bold below). Cookie and an X-SERVER-TOKEN headers are sent with the request, both being the same as those sent with the play.googleapis.com/p/log/batch requests, thus allowing them to be linked together. The Authorization: Bearer header value sent with play.googleapis.com/p/log/batch is included in the response body, see below:

POST https://android.googleapis.com/checkin
Headers:
Cookie: NID=**204=Z-_RXuS4ZdrKlKkBoa...**
User−Agent: Dalvik/2.1.0 (Linux; U; Android 9; Pixel 2 Build/PPR2.180905.005)
X−SERVER−TOKEN: **CAESKQDyi0h8u1NFMSbtIY...**
Request body:f
<...>
2018−09−05\x10\xe8\xc3\xa7\x9d\xa3.2\x0527205:\x0527211B\x06WIFI::H\x00p\x02z\x15\x08\x08\x10\x01\x1a\x0bunspecified"\x00(\x00\x82\x01]
\x0527211\x12\x0c**Tesco Mobile**\x1a\x010 \x00 \x01 \x022\x0f272110103800000:(0AFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFB\x02\x15\xe5\x90\x01\x01\x9a\x01\x04WIFI2\x05en−US8\xfc\xa0\xab\xfb\xd7\xf6\xce\x8d\xc9\x01J\x0c404e36d3f4bdR\x0f**35753708924...**Z\x1a[<**...**>**@gmail.com**]Z\xc7\x02**ya29.a0AfH6SMCv...**
Europe/Dubliniv\xd4\xa3\x7f\x84\xb0\xf7;p\x03z\x1cbfMkwynjHzXGBPc2WT62otR8JkI=\x82\x01\x0c**HT85G1A05...**\x92\x01\x92\\\x08\x03\x10\x01\x18\x01
<...>

The following requests are made less frequently:

POST https://android.googleapis.com/auth
Headers:
device: **3e7736d127071c1a**
app: com.google.android.gms
User−Agent: GoogleAuth/1.4 (walleye PPR2.180905.005); gzip
Request body:
androidId: **3e7736d127071c1a**
app: com.google.android.gms
Email: <**...**>**@gmail.com**
client_sig: 38918a453d07199354f8b19af05ec6562ced5788
Token: **aas_et/AKppINaJgeUW3nalnMjiqfI8KVMmfTkO...g9w=**
callerSig: 38918a453d07199354f8b19af05ec6562ced5788

POST https://www.googleapis.com/experimentsandconfigs/v1/getExperimentsAndConfigs?r=2&c=1
Headers:
X−SERVER−TOKEN: **CAESKQDyi0h8i9ip...**
Authorization: Bearer **ya29.a0AfH6SMAnQ3dP5Z3T...XTQ**
User−Agent: Android/com.google.android.gms/202117028 (walleye PPR2.180905.005); gzip
Request Body:
<...>
0ContactTracingFeature__add_experiment_id_to_logs\x18\x00H\x02\x122
,ContactTracingFeature__advertise_connectable\x18\x00H\x02\x12+
\%ContactTracingFeature__advertise_mode\x10\x01H\x01\x12/
)ContactTracingFeature__advertise_tx_power\x10\x01H\x01\x12:
+ContactTracingFeature__advertise_tx_power_o\x10\xf1\xff\xff\xff\xff\xff\xff\xff\xff\x01H\x01\x129
3ContactTracingFeature__advertisement_metadata_v_1_1\x18\x01H\x02\x12A
;ContactTracingFeature__aggregate_sightings_from_single_scan\x18\x01H\x02\x12H
BContactTracingFeature__allow_downgrade_bluetooth_discoverable_mode\x18\x01H\x02\x12E
?ContactTracingFeature__associated_encrypted_metadata_size_bytes\x10\x04H\x01\x12J
DContactTracingFeature__associated_metadata_encryption_key_size_bytes\x10\x10H\x01\x12K
EContactTracingFeature__ble_scanning_adjust_interval_and_window_values\x18\x01H\x02\x12P
HContactTracingFeature__ble_scheduler_use_scheduled_executor_max_delay_ms\x10\xb0\xea\x01H\x01\x122
,ContactTracingFeature__bt_enable_retry_times\x10\x02H\x01\x126
/ContactTracingFeature__bt_enable_timeout_millis\x10\xc4\x13H\x01\x129
3ContactTracingFeature__calculate_diagnosis_key_hash\x18\x01H\x02\x12G
AContactTracingFeature__check_client_record_after_client_uninstall\x18\x01H\x02\x12:
−ContactTracingFeature__clearcut_sampling_rate \x00\x00\x00\x00\x00\x00\xf0?H\x03\x12O
IContactTracingFeature__completed_matching_request_record_retention_period\x10\x0eH\x01\x12X
AContactTracingFeature__data_quality_attempted_keys_histogram_bins2\x11
\x0f

\x01\x90N\xa0\x8d\x06\xc0\x84=\x80\xad\xe2\x04H\x05\x12>
8ContactTracingFeature__data_quality_log_advertise_failed\x18\x01H\x02\x12B
<ContactTracingFeature__data_quality_log_attempted_keys_count\x18\x00H\x02\x12F
@ContactTracingFeature__data_quality_log_attempted_keys_histogram\x18\x01H\x02\x12C
=ContactTracingFeature__data_quality_log_ble_advertising_stats\x18\x01H\x02\x12M
GContactTracingFeature__data_quality_log_ble_advertising_stats_only_once\x18\x01H\x02\x12?
9ContactTracingFeature__data_quality_log_bluetooth_enabled\x18\x01H\x02\x12G
AContactTracingFeature__data_quality_log_client_app_wake_ups_count\x18\x01H\x02\x12J
DContactTracingFeature__data_quality_log_contains_opportunistic_scans\x18\x01H\x02\x12A
;ContactTracingFeature__data_quality_log_diagnosis_key_files\x18\x00H\x02\x12K
EContactTracingFeature__data_quality_log_diagnosis_keys_provided_count\x18\x00H\x02\x12?
9ContactTracingFeature__data_quality_log_en_module_version\x18\x01H\x02\x129
3ContactTracingFeature__data_quality_log_flag_values\x18\x01H\x02\x12>
8ContactTracingFeature__data_quality_log_location_enabled\x18\x01H\x02\x12:
4ContactTracingFeature__data_quality_log_package_name\x18\x01H\x02\x12=
6ContactTracingFeature__data_quality_log_period_minutes\x10\xf0\x01H\x01\x12?
9ContactTracingFeature__data_quality_log_random_scan_empty\x18\x01H\x02\x12B
<ContactTracingFeature__data_quality_log_random_sighting_rssi\x18\x01H\x02\x12F
@ContactTracingFeature__data_quality_log_rejected_key_files_count\x18\x01H\x02\x12<

6ContactTracingFeature__data_quality_log_rssi_histogram\x18\x00H\x02\x12?
9ContactTracingFeature__data_quality_log_scan_failed_count\x18\x01H\x02\x12A
;ContactTracingFeature__data_quality_log_scan_time_histogram\x18\x01H\x02\x12A
;ContactTracingFeature__data_quality_log_sightings_histogram\x18\x00H\x02\x12?
9ContactTracingFeature__data_quality_log_whitelist_failure\x18\x01H\x02\x12\x7f
7ContactTracingFeature__data_quality_rssi_histogram_bins2B
<...>
∗ContactTracingFeature__partner_public_keys2\x91
\x8e
\x95\x01ch.admin.bag.dp3t:228−v1,MFkwEwYHK...
\x9b\x01lv.spkc.gov.apturicovid:247−v1,MFkwEwYHKoZIzj0...
\xa6\x02it.ministerodellasalute.immuni:222−v1,MFkwEwYH...
\x99\x01fct.inesctec.stayaway:268−v1,MFkwEwYH...
\x98\x01com.covidtracker.hse:272−v1,MFkwEwYHKj...
\x9e\x01uy.gub.salud.plancovid19uy:748−v1,MFkwEwYHKoZI...
\x98\x01de.rki.coronawarnapp:262−v1,MFkwEwYHKoZ...
\x9c\x01de.rki.coronawarnapp.dev:262−v1,MFkwEwYHKoZIzj0...
\x99\x01pl.gov.mc.protegosafe:260−v1,MFkwEwYHKoZI...
\x9f\x01pl.gov.mc.protegosafe.stage:260−v1,MFkwEwYHKo...
\x95\x01sa.gov.nic.tabaud:420−v1,MFkwEwYHKoZIzj0...
\x99\x01au.gov.dta.covidtrace:505−v1,MFkwEwYHKo...
\xb3\x01com.netcompany.smittestop_exposure_notification:238−v1,MFkw...
\x9b\x01jp.go.mhlw.covid19radar:440−v1,MFkwEwYHKoZIzj0C...
\x9c\x01at.roteskreuz.stopcorona:232−v1,MFkwEwYHKoZIzj0...
\xa0\x01gov.vdh.exposurenotification:310−v2,MFkwEwYHKoZIzj0...
\x9b\x01egnc.moh.bruhealthtrace:528−v1,MFkwEwYHKoZIzj0...
\x99\x01com.gha.covid.tracker:266−v1,MFkwEwYHKoZIzj0...
\x8e\x01mx.gob.www:334−v1,MFkwEwYHKoZIzj0CAQYIK...
\x96\x01ec.gob.asi.android:740−v1,MFkwEwYHKoZIzj0...
\x95\x01es.gob.radarcovid:214−v1,MFkwEwYHK...
\x9e\x01hr.miz.evidencijakontakata:219−v1,MFkwEwYHKo...
\x9c\x01br.gov.datasus.guardioes:724−v1,MFkwEwYHK...
\x8f\x01mt.gov.dp3t:278−v1,MFkwE...
\x9f\x01ca.gc.hcsc.canada.stopcovid:302−v1,MFkwEwYHKoZ...
\xa2\x01gov.adph.exposurenotifications:310−v1,MFkw...\x05\x12:
4ContactTracingFeature__require_checkbox_for_clearcut\x18\x01H\x02\x128
2ContactTracingFeature__require_multi_advertisement\x18\x00H\x02\x12A
;ContactTracingFeature__restart_advertising_when_tek_deleted\x18\x01H\x02\x12L
;ContactTracingFeature__risk_score_attenuation_value_buckets2\x0b

XI. Google Play Store Connections

GET https://lh4.googleusercontent.com/MXiDx8ELb7pJl32MDUGr9zufJlk_gwvJYRzyP4WcVx2a7vpj9x57OJxOz00giHKh1pM=rw-h244-v1-e15
Headers:
user−agent: com.android.vending/82071600 (Linux; U; Android 9; en_US; Pixel 2; Build/PPR2.180905.005; Cronet/85.0.4162.4)

GET https://play.googleapis.com/play/log/timestamp
User−Agent: Dalvik/2.1.0 (Linux; U; Android 9; Pixel 2 Build/PPR2.180905.005)

POST https://play.googleapis.com/play/log?format=raw&proto_v2=true
User−Agent: Android−Finsky/20.7.16−all\%20\%5B0\%5D\%20\%5BPR\%5D\%20317546459
Authorization: Bearer **ya29.a0AfH6SMBzxIslH...**

POST https://android.googleapis.com/auth
Headers
device: **3e7736d127071c1a**
app: com.android.vending
User−Agent: GoogleAuth/1.4 (walleye PPR2.180905.005); gzip
Request body:
androidId: **3e7736d127071c1a**
Email: <**...** >**@gmail.com**
client_sig: 38918a453d07199354f8b19af05ec6562ced5788
Token: **aas_et/AKppINaJgeUW3naln...**
callerSig: 38918a453d07199354f8b19af05ec6562ced5788

POST https://www.googleapis.com/experimentsandconfigs/v1/getExperimentsAndConfigs?r=1&c=128
Authorization: Bearer **ya29.a0AfH6SMA...**
User−Agent: Android−Finsky/20.7.16−all\%20\%5B0\%5D\%20\%5BPR\%5D\%20317546459 (api=3,versionCode=82071600,sdk=28,device=walleye,hardware=walleye,product=walleye,platformVersionRelease=9,model=Pixel\%202,buildId=PPR2.180905.005,isWideScreen=0,supportedAbis=arm64−v8a;armeabi−v7a;armeabi) (walleye PPR2.180905.005); gzip

GET https://play-fe.googleapis.com/fdfe/selfUpdate
User−Agent: Android−Finsky/20.7.16−...
x−dfe−device−id: **3e7736d127071c1a**
x−dfe−device−config−token: **CisaKQoTNDUwM...**
authorization: Bearer **ya29.a0AfH6SMB...**
x−dfe−cookie: **EAEYACICSUUyPE...**
x−limit−ad−tracking−enabled: false
x−dfe−phenotype: **H4sIAAAAAAA...**
x−dfe−encoded−targets: **CAESPombgQbxBwLYAwE...**
x−ad−id: **1d2635f5-2af7-4fb3-86e8-5fd6e53f2aff**

POST https://play-fe.googleapis.com/fdfe/bulkPrefetch
user−agent: Android−Finsky/20.7.16−...
x−dfe−device−id: **3e7736d127071c1a**
x−dfe−device−config−token: **CisaKQoTNDUwMTEy...**
authorization: Bearer **ya29.a0AfH6SMBzxIs...**
x−dfe−cookie: **EAEYACICSUUyPENpc...**
x−limit−ad−tracking−enabled: false
x−dfe−device−checkin−consistency−token: **ABFEt1VKH_x3ydZWIzoZSXqkDfbQ...**
x−dfe−phenotype: **H4sIAAAAAAAAADXPS...**
x−dfe−encoded−targets: **CAESPombgQbxBwLYAwE...**
x−ad−id: **1d2635f5-2af7-4fb3-86e8-5fd6e53f2aff**