Analysis Report

The not so private way of tracing contacts: A first analysis of the NOVID20 Android SDK

Michael Roland, Tobias Höller, Michael Sonntag, and René Mayrhofer

Johannes Kepler University Linz
Institute of Networks and Security

Abstract Contact tracing is one of the main approaches widely proposed for dealing with the current, global SARS-CoV-2 crisis. As manual contact tracing is error-prone and doesn't scale, tools for automated contact tracing, mainly through smart phones, are being developed and tested. While their effectiveness—also in terms of potentially replacing other, more restrictive measures to control the spread of the virus—has not been fully proven yet, it is critically important to consider their privacy implications from the start. Deploying such tools quickly at mass scale means that early design choices may not be changeable in the future, and potential abuse of such technology for mass surveillance and control needs to be prevented by their own architecture.

Many different implementations are currently being developed, including international projects like PEPP-PT/DP-3T and national efforts like the "Stopp Corona" app published by the Austrian Red Cross. In this report, we analyze an independent implementation called NOVID20 that aims to provide a common framework for on-device contact tracing embeddable in different apps. That is, NOVID20 is an SDK and not a complete app in itself. The initial code drop on Github was released on April 6, 2020, without specific documentation on the intent or structure of the code itself. All our analysis is based on the Android version of this open source code alone. Given the time period, our analysis is neither comprehensive nor formal, but summarizes a first impression of the code.

NOVID20 follows a reasonable privacy design by exchanging only pseudonyms between the phones in physical proximity and recording them locally on-device. However, there is some room for improvement: (a) pseudonyms should be generated randomly on the phone, and not on the server side; (b) transmitted pseudonyms should be frequently rotated to avoid potential correlation; (c) old records should automatically be deleted after the expunge period; (d) absolute location tracking, while handled separately from physical proximity and only optionally released, can be problematic depending on its use—absolute location data must be protected with additional anonymization measures such as Differential Privacy, which are left to the application/server and may, therefore, not be implemented correctly; and (e) device analytics data, while helpful during development and testing, should be removed for real deployments. Our report gives more detailed recommendations on how this may be achieved.

We explicitly note that all of these points can be fixed based on the current design, and we thank the NOVID20 team for openly releasing their code, which made this analysis possible in a short time window.

Contact: Dr. Michael Roland
@ michael.roland@ins.jku.at
+43 732 2468-4136

INSTITUTE OF NETWORKS AND SECURITY

JOHANNES KEPLER UNIVERSITY LINZ

April 9, 2020 Rev. 1.0

1. Introduction

This report presents the results of a first, incomprehensive code review of the NOVID20 Android SDK. The review is based on the SDK source code published on Github with revision 6532f6f93bcdb28c5b50acc1ebf88126902d72a7[1].

The focus of our review lies on security and privacy. We explicitly neglect bugs that (potentially) disrupt functionality, but otherwise, do not have an impact on privacy. Moreover, we do not try to assess quality of code.

The results published in this report are preliminary and have not yet been confirmed by other researchers.

2. Contact Tracking

The NOVID20 Android SDK uses a collection of multiple techniques to discover other user's devices containing an installation of an app that also uses the NOVID20 SDK: Bluetooth, Bluetooth Low Energy (BLE), and the Google Nearby API (which itself relies, among other technologies, on BLE). In addition, the NOVID20 Android SDK contains unused code relying on the AltBeacon library to scan for various types of BLE beacons.

The central element of contact tracking is a persistent unique user ID. This user ID is generated by a NOVID20 backend server and assigned to each app installation upon initial user registration (see section 3.3). In order to be traceable by others, each device broadcasts this user ID through multiple parallel means. On Android, these are:

- two BLE GATT services (where one[2] is used to identify if other devices have the app installed, and the other[3] contains the user ID as readable data element[4]),
- Google Nearby Messages broadcasts a message consisting of the user ID (also over BLE),
- Google Nearby Connections announces a connection endpoint named with the user ID (using the P2P cluster mode).

At the same time, each device tries to discover the user ID emitted by other devices. On Android, user IDs are discovered through the following methods:

- If the device name of any discovered Bluetooth or BLE device starts with the user ID prefix ("nov20-"), it is used as the user ID of the other device.
- If the other device offers the NOVID BLE GATT service, the user ID value provided by this service is used for the other device.
- If another device connects to the NOVID BLE GATT service, the device name is used as the user ID for that other device (without prefix matching).
- If a message is received through Google Nearby Messages (over BLE), the message is used as the user ID of the other device.
- If a connection endpoint is discovered through Google Nearby Connections, the endpoint name is used as the user ID of the other device.

These discovery mechanisms operate in parallel and continuously record encountered contacts with other devices into a local database.

[1] https://github.com/novid20org/novid20-android-sdk/tree/6532f6f93bcdb28c5b50acc1ebf88126902d72a7
[2] The "app service", UUID 8b9b6576-6db7-11ea-bc55-0242ac130003.
[3] The "NOVID service", UUID b16efb34-6c34-11ea-bc55-0242ac130003.
[4] In a read-only GATT characteristic with UUID 1d45dc00-6db7-11ea-bc55-0242ac130003.

2.1 Contact event records

For each encountered contact, a contact event record is appended to the local contact event storage. The contact event record consists of the following information:

- The user ID received from the other device.
- A timestamp (in milliseconds since the epoch) when the contact occurred.
- The contact tracking mechanism through which the contact was discovered.
- The duration for which the contact was continuously observed. However, this is only used for Nearby Connections. For other mechanisms, a new contact event record is added after some timeout. E.g. for BLE discovery, if another device is constantly seen by BLE discovery, a new contact event would be appended at most every 30 seconds.
- The received signal strength (RSSI) for BLE discovery.

Moreover, two fields named distance and background are present but not used.

2.2 Location tracking

Whenever a contact event occurs, the SDK also collects the current GPS position (though the Android system location services) and stores a location record (consisting of the current timestamp (in milliseconds since the epoch), latitude, longitude, and GPS accuracy). While stored separately, location records and contact event records can be matched based on their timestamp.

Additionally, such a contact record is stored whenever automatic tracking is started (or restarted).

2.3 Automatic tracking

Automatic contact tracking is not always enabled. Instead it is linked to certain user activity (tracked through the Google Activity Recognition API). Specifically, the SDK tries to pause all broadcasting and any active scanning when the device is at rest (not moving) for more than 10 minutes. Automatic tracking is restarted whenever the device starts moving again. Presumably, this mechanism was implemented to improve battery life.

2.4 Expiry of older records

The SDK currently does not provide any means to selectively expire older records (neither for contact events nor for location records). The only option exposed by the NOVID20 Android SDK is a method to wipe the entire database of all contact event and location records.

3. NOVID20 Backend API

The backend API is a REST API (over HTTP/S) used for all communication between an app using the NOVID20 Android SDK and a NOVID20 backend server. It provides endpoints for the following core functions:

1. register a (new) user,

2. update device token,
3. start verification of a user's phone number,
4. confirm verification of a user's phone number with a TAN,
5. report an infection,
6. get infection status, and
7. upload analytics data.

3.1 Request authentication and user identification

Each access to the NOVID20 backend API is subject to authentication with a unique identifier (UID), an SDK access token, and an authentication token. These values are sent to the backend server in HTTP header fields "x-uid", "x-access-token", and "x-auth-token" for every request to the backend API.

The UID and SDK access token are provided by the app that builds upon the NOVID20 SDK. Similarly, strategy for creation and renewal of the authentication token for a given UID is left to the actual app implementation. Hence, no assumptions can be made on how an access token is obtained, generated, or derived.

3.2 Other HTTP headers

Another noteworthy observation with regards to the request headers (sent to the backend API endpoints) is that these headers contain a significant amount of additional information characterizing both the device and its user: the exact Android version and the current locale (language and country code).

3.3 Register a (new) user

This endpoint expects a device token, and returns the user ID used to identify the device (or its user) during contact tracking. Note that user ID and UID are two distinct values.

The device token is provided by the app that builds upon the NOVID20 SDK. Analysis of content and creation of the device token (just like the UID and the authentication token sent with every request) would require access to an actual app that builds upon the SDK. This is out-of-scope for the current analysis. However, it is worth noting that a means to update the device token for a given UID exists.

The user ID is generated by the NOVID20 backend server (as opposed to the UID which is provided by the app implementation). The user ID is used as the primary identifier for the whole contact tracking functionality (cf. section 2). It is a unique value that must start with the prefix "nov20-". Since this identifier is only loaded into the app upon initial user registration, it is a persistent identifier that does not change throughout the whole lifetime of an app installation.

3.4 Report infection

When the app wants to report an infection, the SDK uses this endpoint to send

- the current timestamp,
- a list of all contact event records tracked within the last 72 hours (3 days), and

- (optionally) a list of all location records ever recorded by the app (i.e. a GPS track).

to the NOVID20 backend server.

It is up to the app to choose if the list of GPS locations should be included. Interestingly, there is no 72 hour limit imposed on the location information. If an app implementation chooses to also upload location records together with the contact events, the complete list of all location records ever tracked by the app is uploaded. Since a new location record is stored whenever scanning is enabled and, in particular, whenever a contact event is recorded, this implies that the server receives timestamps for all contact events ever recorded by the app.

In order to verify the validity of a report, backend API provides functionality to verify a user's phone number (using one endpoint to trigger sending of a verification code (TAN) to the phone number, and a second endpoint to validate a received TAN). It should be noted that TAN verification automatically triggers reporting of an infection. An app can, however, also trigger reporting of an infection without a preceding phone number verification. An analysis of the backend server implementation would be necessary to determine if phone number verification and infection reporting are decoupled from each other.

3.5 Get infection status

This endpoint returns the user ID, the current infection status of the user, and the number of infected contacts. Consequently, it can be used to check if the user had previously reported an infection and if the user's user ID had been reported as a recent contact by another (infected) user.

3.6 Upload analytics data

This endpoint is used to regularly upload analytical data to the backend server. The data contains, among other items, the device model (brand name, model identifier), a list of internal events of the SDK (such as when automatic contact tracking was switched on or off), and events generated by the app implementation.

As with all requests to API endpoints, the upload of analytical data contains authentication headers that permit identification of the user and/or device based on the unique identifier. Consequently, analytical data could be linked to a specific user.

4. Privacy Issues

With regard to privacy, the NOVID20 association (NOVID20 - Technologieplattform zum Krisenmanagement, ZVR 1338183890) claims (https://www.novid20.org/en):

> *"Privacy is one of our core values. Our solution is built around it, thus we do not collect personal data like your name or your address. Everything is stored locally and encrypted on the device - you have full control!"*

They further state (https://www.novid20.org/en/our-solution):

> *"We work together with high-tier data-security companies and data-protection lawyers to ensure the best level of privacy. Most valuable is the fact that we only store data on your own device in encrypted form. Therefore, we can guarantee that the users stay anonymously and no risk stigmatisation of infected people emerges. The App only needs permission for all phones features to trace your proximity contacts, but this data is not exported to any authorities or foreign servers. The Audio recording is necessary to detect nearby phones based on high-frequency sounds.*
>
> *We make sure your privacy is treated the right way."*

This, together with the fact that the source code for the NOVID20 SDKs (for both Android and iOS) was made publicly available[5], sounds very promising. Unfortunately, our analysis of the NOVID20 Android SDK revealed that most of these promises are not fulfilled by the current implementation.

Nevertheless, we want to point out that publication of the source code was an important step into the right direction. Only this made a thorough analysis by independent third parties possible in the first place (resulting in our report and even suggestions for improvement). With a closed-source solution (like it is currently the case for the "Stopp Corona" app of the Austrian Red Cross), such an in-depth analysis would not easily be possible, leaving similar privacy threats uncovered.

4.1 Data retention

The SDK is designed to only upload the contact events of the last 72 hours (three days) to the server when an infection is reported. This value is hard-coded into the SDK. No option is provided to the app implementation to modify that time span.

However, despite limiting the upload time frame, older contact events are never expired/purged from the internal database. There is only an option to delete all data from the app, which is useless since that would also wipe the data relevant for reporting an infection.

Retaining unnecessary information is, by definition, privacy unfriendly.

4.2 Location tracking

Besides the contact events themselves, the SDK also tracks the GPS location for every contact event and upon certain app state transitions (but only if the user granted the corresponding Android permission[6]). The SDK can be configured to upload this information to the server when reporting an infection.

However, as only location data of (confirmed) infected users is uploaded, the benefit of this option seems questionable. This additional data would allow building movement profiles of infected individuals. Non-infected users cannot query the server for information where (and when) infected individuals were observed. Also, they do not upload their own location data to the backend server (since this could only happen when reporting your own infection). Thus, the locally stored GPS track is of no use in determining infection status.

With regards to the principle of data minimization, if the user decides not to upload the data when reporting an infection, there is no reason to collect that in the first place.

Overall, we believe that uploading GPS tracks, and particularly the fact that GPS data can easily be mapped to all reported contact events (through the timestamp), reveals too much information about users' private lives to a central entity, without a significant benefit. Location information could even be used to discern violations of curfew regulations. Whether this is necessary or useful for quarantine measures is doubtful.

4.3 Persistent user identifier

Every user is assigned a static long-lived user ID (a pseudonym), which is constantly broadcast via BLE and Google Nearby during typical app operation (i.e. when performing contact

[5] Published on Github at https://github.com/novid20org/ (even under the open-source license "GNU General Public License v3.0").

[6] Note that the SDK reports the fact that this permission (`ACCESS_FINE_LOCATION`) was not granted as an error condition to the app implementation.

tracking). This enables anyone in proximity of an individual (app user) to discover that user's persistent user ID. This is not only limited to legitimate other app users. Instead, anyone with equipment to scan for Bluetooth devices (in the case when the Bluetooth device name is used to carry the user ID) or to read BLE GATT characteristics (in the case when the NOVID GATT service is used to carry the user ID) could perform that discovery. Consequently, users can easily be deanonymized (by mapping the user ID back to an actual person) and can easily be re-identified and tracked in future.

The backend server can also use lists of contact events with persistent user IDs to count and categorize interactions of users to a degree far beyond what would be necessary for quarantine measures.

4.4 Targeted quarantine

Since user IDs are persistent and global, it may be possible for a malicious actor to discover and forge user IDs in order to mount attack scenarios where a forged infection chain (e.g. by spoofing user IDs in the proximity of a known infected individual) would lead to wrong assumptions about contact with infected persons. In a worst-case scenario, this could be abused to force a person into being quarantined.

4.5 Timestamps

Both contact events and location information carry timestamps with a resolution of milliseconds. While there might be some benefit of holding and evaluating that information locally (e.g. for the purpose of characterizing different types of encounters with other individuals), we see absolutely no reason to provide this information to a centralized entity (backend server). When an infection is reported, only the contacts of the last three days are transmitted anyway. There is no reason for the server to know the exact timestamp of when another individual was met.

Timestamps should either not be sent to the server or, at the very least, be significantly reduced in precision.

4.6 Transmission of complete contact event list

In fact, we do not see a benefit in transmitting details contact event logs to a central server at all. The app could easily perform categorization of different types of encounters (e.g. a person just walking by vs. a person staying in the same room for a longer period) locally on the collected data. The app could then perform selective and contact-time independent checking of user IDs.

4.7 Activity tracking

The SDK relies on the Google Activity Recognition API to identify if a phone is moving or at rest. When at rest for a longer time, the phone stops broadcasting its own ID and also stops searching for contacts (presumably to preserve battery). While this is a good idea in theory, it comes at the price of a massive privacy implication: The SDK is designed to collect analytics data whenever discovery is started or stopped. The SDK typically reports analytics data to the backend server on an hourly basis. This allows the server to track all transitions between the phone being at rest and the phone moving. This information can be used to infer how much time each user spends at rest, giving a deep inside in an individual's daily life. The fact that this information is sent continuously (and not only when an infection is reported) constitutes a massive privacy issue.

4.8 Analytics in general

As we do not have access to an actual app implementation using the NOVID20 Android SDK, not all details about potential analytics are available to us. However, besides the analytics sent to the NOVID20 backend itself, the SDK is deeply integrated with Google Firebase Analytics[7] and also contains code that suggests (future?) integration with Customer.io behavioral tracking[8].

All events collected by the SDK's own analytics are also made available to Firebase Analytics. This includes starting and stopping contact tracking. As this correlates to user activity (start/stop moving), this provides Google and the app provider with the same detailed user activity log as provided to the NOVID20 backend (see section 4.7 for the privacy implications). Additionally, events like when the user switches between different screens can be reported. How that additional analytics is actually used is beyond the scope of the NOVID20 Android SDK and is left up to the app implementation.

Besides the actual analytics data, transmission of that data also provides connection metadata (such as the IP address of the device). Such connection metadata is not only available to the actual data recipient (Firebase, Customer.io, NOVID20 backend) but also to any intermediary (e.g. the provider).

The internal analytics of the SDK is synchronized with the NOVID20 backend on an hourly basis. This transmission is even authenticated with the UID and the authentication token. Technically, this allows the backend to easily link analytics data to specific users.

None of the embedded analytics are essential (or even necessary) for the core functionality of the SDK (contact tracking). Potentially, collected data may even be transferred and stored outside the EU. The collected analytics allow to deduce further data about users and increase the chance to identify them. Regarding privacy by design, this is a severe failure and completely unnecessary—except, perhaps, from a commercial viewpoint of monetizing collected user data.

5. Possible Improvements

We would like to point out that there is already a widely accepted agreement on how privacy-preserving proximity tracing could be achieved, commonly known as Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT)[9]. Most of the suggestions made in this section can be directly derived from the PEPP-PT recommendations.

5.1 Dynamic temporary user ID

Replace the persistent user ID with a dynamic, time-limited identifier. That makes impersonation and correlation of data much harder, without impacting an app's capability to track and notify contacts. This would work best, when confirmed infected users reveal their own set of identifier used during the observation time frame, instead of reporting all their contact encounters.

5.2 Alert functionality

At the moment, a central entity (NOVID20 backend) is responsible for maintaining a global view of all encounters with infected individuals and for making all decisions. The central

[7]https://firebase.google.com/products/analytics
[8]https://customer.io/
[9]https://www.pepp-pt.org/content

server decides when an epidemiologically relevant contact had been made and the server decides who gets notified.

In line with the dynamic temporary user IDs, the system should be switched to a scheme where the client can request a list of infected dynamic user IDs and check if any of these user IDs is found in the local contact event database. That way, no information about actual contact events would ever become available to a central entity. This would of course compromise the privacy of infected individuals if exact time and location information was stored by a device. This is a clear argument against storing such additional (and unnecessary) information and keeping the information tracked about individual contact events at a minimum.

Overall, this approach distributes the power across all the app users instead of concentrating everything at one central instance.

5.3 Analytics data

Analytics of user behavior is not necessary at all for the application. Dropping analytics entirely should be considered and is the preferred approach. If statistics are believed to be unavoidable, the analytics data must not directly link to individual users. Hence, user authentication (UID and authentication token) must not be included in requests to analytics collection endpoints.

Furthermore, we believe that sending analytics data to both an external provider and the NOVID20 backend is excessive.

5.4 Data reduction

Currently the NOVID20 Android SDK records and transmits far too much information. The entire project should be reviewed to make sure that only absolutely necessary information is stored and shared. A few examples have already been identified and listed in the preceding sections (e.g. location, timestamps, etc.)

5.5 Caching of MAC address to user ID mapping

Currently, the SDK assumes that a Bluetooth MAC address observed for a specific user ID will always map to the same user ID whenever that MAC address is (later) observed again. Cached entries are maintained for the whole lifetime of the application process. This can lead to an issue with MAC address randomization: When two different devices generate the same randomized MAC address, the second of those devices will be assume to have the user ID of the first device.

While it is very unlikely that two devices in proximity generate the same MAC address simultaneously, it is a lot more likely that two different devices are observed by one device at two distinct moments in time with the same MAC address. If the cache is kept for a long time (days, maybe even hours) it is not unlikely that two different devices are discovered by the same random MAC address.

This could easily lead to contact events with infected individuals going undetected (when an infected person is discovered by the user ID of a non-infected person) or misclassification of infected individuals beyond the N hours time span (when a non-infected person is discovered by the user ID of an infected person that was actually met more than N hours ago).

5.6 TAN verification

This does not really pose a privacy issue but a potential security issue. Currently, the endpoints for reporting infections and validating TANs are independent of each other. When designing the server side, special care must be taken that reporting infections is only possible by legitimate users (i.e. those with a verified phone number). This requires a stateful API, which keeps track of which users have submitted a valid TAN. It would be preferable tightly link verification and reporting by verifying the TAN directly though the infection reporting endpoint, so that a user has to provide a valid OTP every time they want to report an infection.