UNMASKED II

An Analysis of Indonesia and the Philippines' Government-launched COVID-19 Apps

By Pellaeon Lin, Jeffrey Knockel, Irene Poetranto, Stephanie Tran, Justin Lau, and Adam Senft

DECEMBER 21, 2020
RESEARCH REPORT #136

Copyright

Suggested Citation

Pellaeon Lin, Jeffrey Knockel, Irene Poetranto, Stephanie Tran, Justin Lau, and Adam Senft. "Unmasked II: An Analysis of Indonesia and the Philippines' Government-launched COVID-19 Apps," Citizen Lab Research Report No. 136, University of Toronto, December 2020.

Acknowledgements

Research undertaken in this report was supervised by Professor Ronald J. Deibert, principal investigator and director of the Citizen Lab.

Financial support for this research has been provided by the John D. and Catherine T. MacArthur Foundation, the Ford Foundation, the International Development Research Centre, and Sigrid Rausing Trust.

About the Citizen Lab, Munk School of Global Affairs & Public Policy, University of Toronto

The Citizen Lab is an interdisciplinary laboratory based at the Munk School of Global Affairs & Public Policy, University of Toronto, focusing on research, development, and high-level strategic policy and legal engagement at the intersection of information and communication technologies, human rights, and global security.

We use a “mixed methods” approach to research that combines methods from political science, law, computer science, and area studies. Our research includes investigating digital espionage against civil society, documenting Internet filtering and other technologies and practices that impact freedom of expression online, analyzing privacy, security, and information controls of popular applications, and examining transparency and accountability mechanisms relevant to the relationship between corporations and state agencies regarding personal data and other surveillance activities.

Contents

To accompany this report, the authors have written an FAQ. The FAQ and the key findings of this report has been translated to Tagalog and Bahasa Indonesia.

Key findings

- **As part of the Citizen Lab's research into the security and privacy of applications, we report on issues we discovered with three COVID-related applications in Indonesia and the Philippines.**
- **PeduliLindungi**, a COVID-19 contact tracing app launched by the Indonesian government, collects and associates users' geolocation coordinates with their name, phone number, and device identifiers. It also collects a user's WIFI MAC address and local IP address, which are not necessary for the app's main features to function.
- **StaySafe PH,** a COVID-19 contact tracing app launched by the government of the Philippines, collects device geolocation data and stores it in an insecure manner. Through a vulnerability in the backend database used by this app, we were able to access the geolocation data of hundreds of thousands of users. We are concerned, but did not confirm, that it would be possible to use users' movement patterns to deanonymize them and to discover their health status.
- We disclosed this vulnerability to Multisys, the developer of StaySafe PH, who released updates for the Android and iOS versions of the app and secured the platform's backend database in response.
- **COVID-KAYA**, a COVID-19 case tracking platform used by healthcare workers in the Philippines, requests access to sensitive private information that does not appear to be necessary for the app's principal functions. We previously reported that both the Android and web versions of this platform contained vulnerabilities that would have allowed unauthorized users to access private data about the app's users and potentially patient data.

Introduction

In July 2020, we wrote a short blog post regarding COVID-19's impact on marginalised communities in Indonesia, the Philippines, Singapore, and South Korea. Following this publication, we conducted an analysis of COVID-19 apps launched by the governments of Indonesia and the Philippines because of reported concerns over the collection of personal data through the apps (see Detik.com and INQUIRER.net for Indonesia's and the Philippines' apps, respectively), as well as previous incidents of COVID-19-related data breaches in Indonesia and the Philippines.

In this report, we analyze Indonesia's PeduliLindungi app, as well as the Philippines' StaySafe PH and COVID-KAYA apps. In particular, we analyzed Android versions of Indonesia's PeduliLindungi app (version 2.2.2), as well as the Philippines' StaySafe PH app (version 0.12) and COVID-KAYA app (version 1.4.7). We chose to analyze the Android versions of these apps because Android is the predominant mobile operating system used in the Southeast Asian region.

Our analysis focuses on evaluating the permissions *declared* by the Android version of each app, particularly on what Google calls "dangerous permissions" (or permissions that "access private user data, a special type of restricted data that includes potentially sensitive information"). Examples of dangerous permissions include access to a device's photos, camera, location, contacts, calendar, phone status, and other access to sensitive user information. Understanding which permissions an app requests is important because the Android operating system restricts apps' access to sensitive system functionalities based on which permissions an app has been granted.

Technical Methodology

To understand the purpose of each permission that each app requests, we used multiple methods, including surveying news media, reviewing apps' privacy policies and the claims of the apps' developers, reviewing previous literature analyzing these apps, as well as reverse engineering the apps' code and analysing the apps' network traffic. For each dangerous permission declared, we then evaluated whether the permission was necessary for the functionality of the app.

In Android's permission system, apps must declare the permissions that they use in a manifest file ("AndroidManifest.xml") included in the app package. From the manifest's list of permissions, non-dangerous permissions are requested upon installation of the app. Dangerous permissions must be requested using the "runtime permissions" model introduced in Android version 6. Runtime permissions are requested through a dialogue interface, typically when the permission is first required to be used. Only after the user grants the runtime permission is the app allowed to use that permission. Usually, when users deny runtime permission requests, apps will continue to function without the features that needed these permissions.

Note that it is possible that an app *declares* a runtime permission in its manifest, but never *requests* such permission through the runtime permission dialog. In such cases, these permissions are in effect not granted. Even if an app has requested and has been granted a permission, this does not mean that the app will use the system functionalities protected by that permission, only that the app *can* use those functionalities. In this report, we refer to any permission that is either declared but never requested, or requested but never used, as an *unused* permission. Unused permissions themselves do not pose serious security and privacy risks. However, they do present a threat if the app is compromised by an attacker, as an attacker who acquires the capability of remotely executing code in the app would additionally acquire the unused permission.

When reverse engineering each app's code, in order to find the developers' intention behind requiring each permission, we took the following *static analysis* approach:

1. Search for the permission's name in the source code to find code segments that check for or request that permission.
2. Find the callers of these code segments to see under what conditions the permission is checked or requested.
3. From these callers' code contexts, determine what high level operation the callers are trying to perform and the high level feature that the code is implementing.

When analyzing each app, we also used the following *dynamic analysis* methods:

1. Since dangerous permissions must be requested at runtime, we ran each app to see which declared permissions were actually requested.
2. When we ran each app, we also captured the network traffic data that it transmitted to see what sensitive information was included.

To capture network traffic, we ran the app on a "rooted" Android system with a custom SSL Certificate Authority certificate installed. This setup allows us to intercept and decrypt SSL-encrypted network traffic. "Rooting" is a kind of after-market modification to Android phones, which allows users to directly control the devices using the "root" superuser account present in all Android systems.

There are some limitations to this methodology:

1. With our static analysis approach, we may not be able to find use of a permission if the developer intentionally obfuscates the source code. Sometimes static analysis can be complemented by observing an app's runtime behaviour. However, observing an app's runtime behaviour (dynamic analysis) also has limitations.
2. With our dynamic analysis approach, our running environment may not satisfy the

conditions for when the app requests a permission or transmits certain information. For example, COVID-KAYA requires users to sign in before they can use its features. However, we do not have the required credentials to do so. Thus, we can only observe the app's behavior from launch to the login screen.

As with any reverse engineering approach, the approach we took to analyze these apps may only provide a partial view of the apps' complete behaviour.

Indonesia's PeduliLindungi app

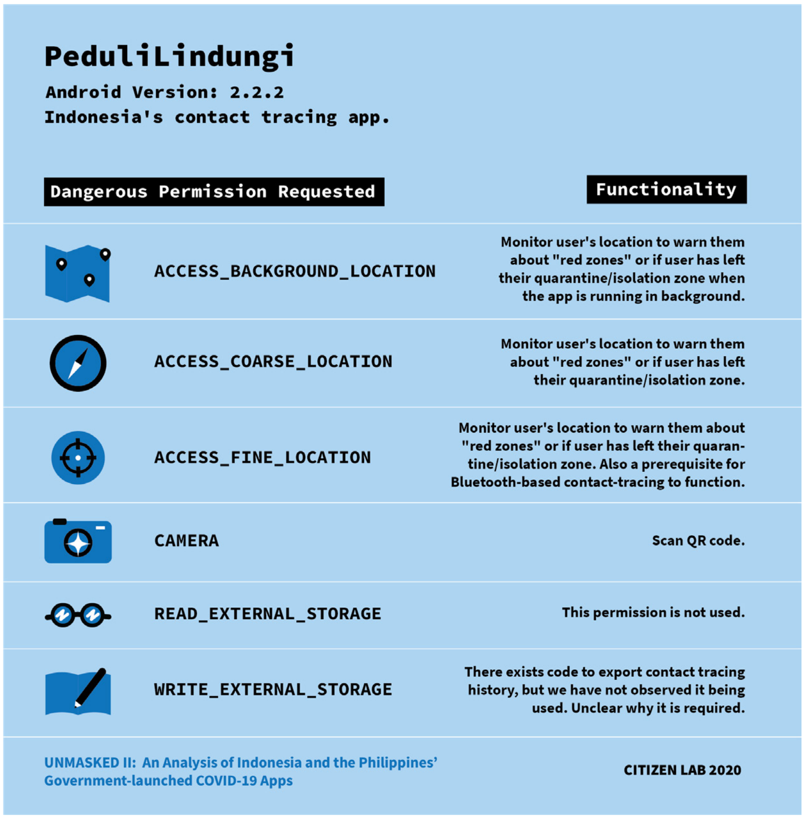
In this section, we analyze the dangerous permissions used by Indonesia's PeduliLindungi app.

About PeduliLindungi

The PeduliLindungi app was launched at the end of March 2020 by Indonesia's Ministry of Communication and Information Technology (MCIT) and the Ministry of State-Owned Enterprises (MSOE) to track exposure to COVID-19. According to PeduliLindungi's in-app interface messages, it uses Bluetooth to periodically scan for nearby Bluetooth devices, similar to Google and Apple's Exposure Notification System. The app's interface messages also state that by providing location access, users will be notified when they are in red zones, defined as areas with positive COVID-19 cases (confirmed and presumptive), and for users who are in independent quarantine status, they will be notified if they have left their quarantine/isolation zone.

The MCIT has continued to add new features to the app, including a "digital diary" function that records a list of the locations visited by the user, and a facial recognition function to measure the user's temperature and to determine whether a mask or face covering is on before the user enters a public space or building. Furthermore, a privacy policy was added to the PeduliLindungi website after privacy and security concerns were aired by rights advocates, including by 13 organizations who wrote an open letter to the MCIT.

Our analysis found that the app declares multiple dangerous permissions, including location permissions capable of recording geolocation, camera permissions capable of taking photos and recording video, as well as device storage permissions capable of reading users' photos and other files. In the remainder of this section, we examine if and how these permissions are actually being used by the app.



PeduliLindungi

Android Version: 2.2.2

Indonesia's contact tracing app.

Dangerous Permission Requested	Functionality
ACCESS_BACKGROUND_LOCATION	Monitor user's location to warn them about "red zones" or if user has left their quarantine/isolation zone when the app is running in background.
ACCESS_COARSE_LOCATION	Monitor user's location to warn them about "red zones" or if user has left their quarantine/isolation zone.
ACCESS_FINE_LOCATION	Monitor user's location to warn them about "red zones" or if user has left their quarantine/isolation zone. Also a prerequisite for Bluetooth-based contact-tracing to function.
CAMERA	Scan QR code.
READ_EXTERNAL_STORAGE	This permission is not used.
WRITE_EXTERNAL_STORAGE	There exists code to export contact tracing history, but we have not observed it being used. Unclear why it is required.

UNMASKED II: An Analysis of Indonesia and the Philippines' Government-launched COVID-19 Apps

CITIZEN LAB 2020

Figure 1: Dangerous permissions requested by PeduliLindungi and their functionality.

Analysis of PeduliLindungi's dangerous permissions

1. Location

We found that the app requests multiple location permissions, including ACCESS_BACKGROUND_LOCATION, ACCESS_COARSE_LOCATION, and ACCESS_FINE_LOCATION. The latter two permissions allow the app to obtain location information from the operating system, and the first permission allows the app to do so even in the background. In effect, this means that the app can track the device's physical geolocation at all times.

According to the app's interface message, location information is used to notify users when they enter and stay in crowded areas or "red zones" for more than 30 minutes. "Red zones" (also referred to as "affected zones") are areas that have recorded COVID-19 positive cases or have individuals under quarantine. PeduliLindungi's privacy policy expounds on this, saying that the app collects location data to "support the 'COVID-19 Zoning' feature which keeps running without the user opening the application. Users will receive notifications when moving from one zone to another. The application will provide information whether the zone where the user is in is a red zone (infected zone), green zone (safe zone) or yellow zone (alert zone) affected by the COVID-19 virus." The privacy policy also states that user notifications are sent in 'real time,' even without the user opening the app.

By analyzing the app's network traffic, we confirmed that PeduliLindungi sends the device's geolocation coordinates along with device identifiers to two endpoints: (1) https://api.pedulilindungi.id/zone/v2 and (2) https://oat.udata.id/addTrackingPLLive.php. We found that the first endpoint is used to determine whether the user is currently in any COVID-19 red zone based on their current geolocation coordinates. The second is an analytics endpoint hosted by PT Telekomunikasi Indonesia Tbk. (Telkom Indonesia), a telecommunications company that is majority owned by the Indonesian government and is the app's developer. In addition to a user's geolocation and WIFI MAC address, each user's full name and phone number are also sent to the second endpoint.

Neither of these data transmissions are essential for contact tracing. Sending geolocation coordinates to a central server is not necessary for contact tracing because the app could, for example, use Bluetooth to detect nearby PeduliLindungi users, including those who are COVID-19 positive, and to identify crowded areas. An app implementing contact tracing in this manner would not need to maintain centralized lists of red zones. Furthermore, the data items sent to the Telkom Indonesia analytics endpoint, including a user's geolocation, device identifier, full name, and phone number, serve no clear purpose in protecting users from COVID-19. It is unclear from PeduliLindungi's privacy policy that these data items are sent to Telkom Indonesia, how they are used by Telkom Indonesia, and whether they are used for digital advertising.

2. Camera

We investigated how the camera permission was being used by the app because this permission can potentially be used to take photos of or record videos of users. By analyzing news media's description of the app's features, we found that use of the camera permission could be explained by two different features of the app. The first feature is a QR code scanner. This feature is used for the "digital diary" function that records a list of the locations visited by the user, particularly when the user scans government-provided QR codes. For example, when Indonesian citizens and foreign nationals enter into the country. The second feature is facial recognition technology, which is used to measure the user's temperature and to determine whether a mask or face covering is on before the user enters a public space or building.

3. Device storage

We found that the app declares two permissions to access external storage, READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE. External storage on Android refers to the storage space that can be directly accessed by the user (usually to store user files, such as documents and media) and is shared among all apps. If the user grants the PeduliLindungi app access to external storage, it has access to files created by the user or other apps, including potentially sensitive files, stored in that storage. On Android 10 and above, the implications of such access is reduced by the newly introduced “scoped storage” system feature, which reduces the scope of the files that apps can access on external storage.

By reverse engineering PeduliLindungi, we found that the app includes code to periodically export its local Bluetooth contact history to external storage as a file named “pedulindungi.json” (sic). However, we did not observe this behaviour when actually trying to use the app, which may be because our testing conditions did not trigger such a backup. This finding can explain why WRITE_EXTERNAL_STORAGE permissions were declared.

It is unclear why the app must export such information to external storage. If this export feature is not used, the app’s developers can cease declaring the WRITE_EXTERNAL_STORAGE permission, as this provides no benefit to the app’s functionality. Moreover, by storing Bluetooth contact data in external storage, other apps with external storage permissions would also have access to read this data in versions of Android earlier than 10.[1]

On the other hand, READ_EXTERNAL_STORAGE was never requested, and, outside of declaring the permission, there was no mention of it in its source code. The developers can cease declaring this permission.

Other issues

When inspecting PeduliLindungi’s source code, we found that it contains a “root” detection library called RootBeer. When running the app using a rooted phone, it only showed a message saying rooted devices are not supported. The app was otherwise unusable. Root detection is a common method developers use to make reverse engineering more difficult. We had to employ an additional layer of system modification with Frida and a root detection bypass script in order to use the app on a rooted phone.

1 Android 10 introduced the “scoped storage” system feature, which limits apps’ access to external storage.

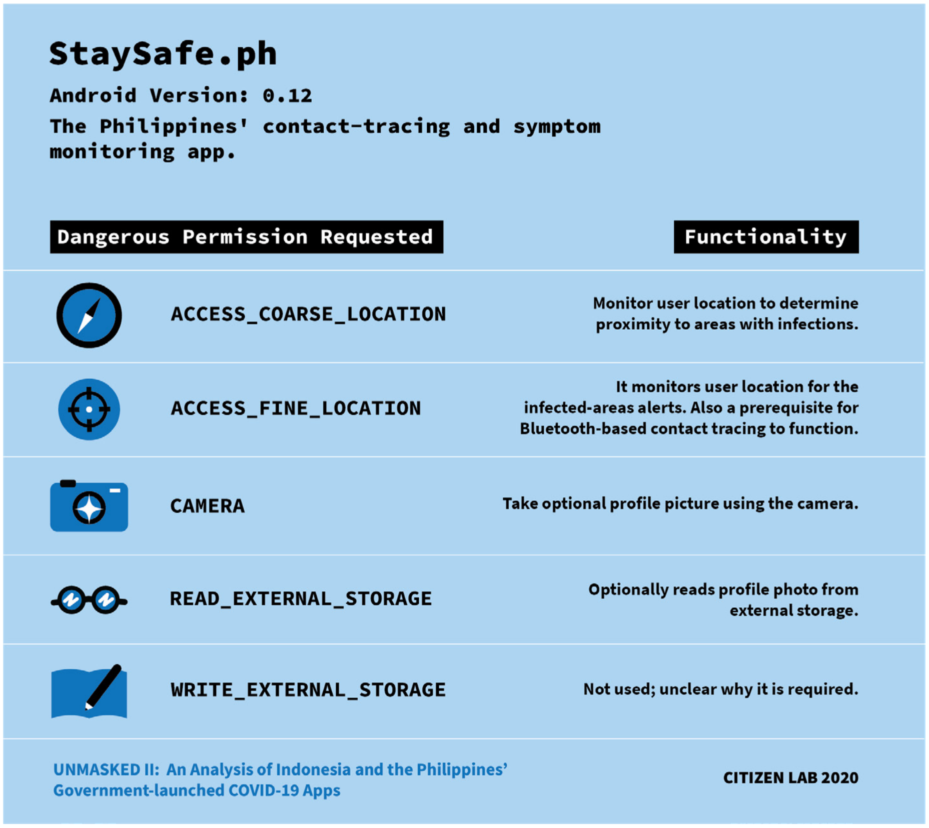
The Philippines' StaySafe PH and COVID-KAYA apps

In addition to the Indonesian PeduliLindungi app, we also analyzed the permissions used by two Philippine apps, StaySafe PH and COVID-KAYA.

About StaySafe PH

StaySafe PH is the Philippines' officially adopted contact tracing and symptom monitoring app. It was developed by Multisys Technologies Corporation, a Philippine software solutions firm. StaySafe PH has a "privacy notice" that outlines the types of information collected, the purposes of data collection, and how that information is used, shared, and retained.

Like with PeduliLindungi, we found that the app requests multiple dangerous permissions, including location permissions capable of recording geolocation, camera permissions capable of taking photos and recording videos, as well as device storage permissions capable of reading users' photos and other files. In the remainder of this section, we examine if and how these permissions are actually being used by the app.



StaySafe.ph

Android Version: 0.12

The Philippines' contact-tracing and symptom monitoring app.

Dangerous Permission Requested	Functionality
ACCESS_COARSE_LOCATION	Monitor user location to determine proximity to areas with infections.
ACCESS_FINE_LOCATION	It monitors user location for the infected-areas alerts. Also a prerequisite for Bluetooth-based contact tracing to function.
CAMERA	Take optional profile picture using the camera.
READ_EXTERNAL_STORAGE	Optionally reads profile photo from external storage.
WRITE_EXTERNAL_STORAGE	Not used; unclear why it is required.

UNMASKED II: An Analysis of Indonesia and the Philippines' Government-launched COVID-19 Apps

CITIZEN LAB 2020

Figure 2: Dangerous permissions requested by StaySafe PH and their functionality.

Analysis of StaySafe PH's dangerous permissions

1. Location

We found that the app requests two location permissions, ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION, that are used to implement two features.

First, StaySafe PH uses these permissions to implement contact tracing. By analyzing reports from news media, we found that StaySafe PH's contact tracing function relies on the exchange of Bluetooth identifiers. Using Bluetooth in this manner does not immediately reveal the user's location, but it can indirectly do so by revealing which other Bluetooth devices the user has been nearby.

Second, StaySafe PH uses location privileges for its notification system that alerts users when they are in areas with probable, suspected or confirmed COVID-19 cases. News media reported that StaySafe PH collects the user's GPS location, which is required for the notification system to function. StaySafe PH's privacy notice corroborates this data collection as it lists geolocation data (if geolocation is enabled) among the types of information that the app may obtain. David Almirol Jr., the CEO of Multisys, has acknowledged that the app can function without collecting geolocation data, but it would require the removal of features such as area notifications.

Our analysis confirmed that StaySafe PH collects users' geolocation coordinates and that they are uploaded and stored on a Google-hosted Firebase Database. This central storing of user data is not necessary for digital contact tracing, as evidenced by protocols such as DP-3T and Apple and Google's Exposure Notification System. Demonstrating the risk in centrally collecting location information, we found a security vulnerability in the Firebase database that collected location information. This vulnerability allowed attackers to easily track physical locations of all StaySafe PH users, which we explain in more detail below ("Vulnerabilities discovered in StaySafe PH").

2. Camera

Through reverse engineering, we were able to identify that the app requests permission to use the camera to allow users the option of taking a profile picture. From the app's network traffic, we confirmed that the profile picture is uploaded to the server at https://photo.staysafe.ph/. However, the interface did not explain how the profile picture would be used. Although taking a profile picture was optional, apps should always clearly communicate how collected personal data will be used, yet StaySafe PH failed to do so.

3. Device storage

The app declares two permissions related to external storage, READ_EXTERNAL STORAGE and WRITE_EXTERNAL STORAGE. The READ_EXTERNAL STORAGE permission is used by StaySafe PH to select an optional profile photo. The app also declares the WRITE_EXTERNAL STORAGE permission, which is used to write to a device's external storage. However, the purpose of this permission is unclear because we did not find it being used by the app.

Vulnerabilities discovered in StaySafe PH

We discovered a vulnerability in the backend database used by StaySafe PH that allowed us to access all of the information contained in that database. As the database contains a list of user identifiers and their exact geolocation coordinates, an attacker could track the physical locations of hundreds of thousands of users.

The vulnerability in Stay Safe PH's backend database stems from improper authorization of the database. Even though non-logged-in users were denied access to the database, logged-in users were improperly granted global read access and potentially write access. This means that the full contents of this database, while not being entirely public, can still be accessed easily by registering an account. This access should not be available to ordinary users of the app and should be restricted to special users of the database management system.

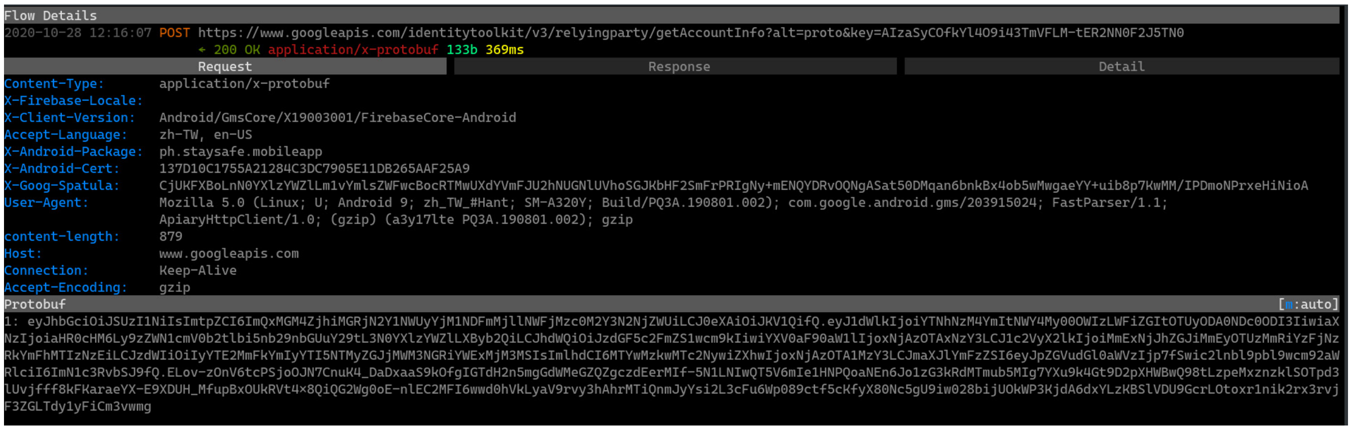
Other than user geolocation information, the leaked database also contains each user's universally unique identifier (UUID). UUIDs are 128-bit numbers used to identify information in computer systems. In StaySafe PH, UUIDs are used to identify individual users internally, instead of using usernames. In StaySafe PH, as with most apps, the UUID corresponding to a user does not change over time, and, therefore, an attacker may persistently track a user's location by tracking the coordinates associated with the user's UUID. Using a separate API at https://ws.staysafe.ph/api/v1/me/trace-registrants, we found that we can also query a user's self-reported health status (normal, having symptoms, etc.) by UUID.

In StaySafe PH, we have not found a way to look up a user's real-world personally identifiable details (such as names) by their UUID. However, a user's identity in the database could be inferred by their changing location information. For example, if an attacker knew the victim's home and work location or other movement patterns, they could find the victim's UUID by searching for the victim's changing geolocation from the geolocation coordinates of all users. Once the victim's UUID is determined, we are concerned that their health status could be easily tracked using the API to track users' health status described above.

Information stored in this backend database appears to be used for StaySafe PH's "Protect Me" feature, which shows the health statuses of nearby users and is intended to display any nearby suspected, probable, or confirmed cases.[2] As we have suggested to Stay Safe PH's developer, limitations should be placed on querying this database, and the data itself should be anonymized. Given their privacy impact, features that require retrieving other users' geolocation coordinates should be reevaluated.

Exploitation

By capturing its network traffic, we observed StaySafe PH accessing a Firebase Database at https://staysafe-prod.firebaseio.com/. Firebase is a suite of cloud services owned by Google that provides backend functions for mobile applications. We followed a methodology developed by Brandon Evans to retrieve the user's authentication token to the Firebase database.



```
Flow Details
2020-10-28 12:16:07 POST https://www.googleapis.com/identitytoolkit/v3/relyingparty/getAccountInfo?alt=proto&key=AIzaSyCOfkYl4O9i43TmVFLM-tER2NN0F2J5TN0
                         ← 200 OK application/x-protobuf 133b 369ms
                         Request                              Response                              Detail
Content-Type:        application/x-protobuf
X-Firebase-Locale:
X-Client-Version:    Android/GmsCore/X19003001/FirebaseCore-Android
Accept-Language:     zh-TW, en-US
X-Android-Package:   ph.staysafe.mobileapp
X-Android-Cert:      137D10C1755A21284C3DC7905E11DB265AAF25A9
X-Goog-Spatula:      CjUKFXBoLnN0YXlzYWZlLm1vYmlsZWFwcBocRTMwUXdYVmFJU2hNUGNlUVhoSGJKbHF2SmFrPRIgNy+mENQYDRvOQNgASat50DMqan6bnkBx4ob5wMwgaeYY+uib8p7KwMM/IPDmoNPrxeHiNioA
User-Agent:          Mozilla 5.0 (Linux; U; Android 9; zh_TW_#Hant; SM-A320Y; Build/PQ3A.190801.002); com.google.android.gms/203915024; FastParser/1.1;
                     ApiaryHttpClient/1.0; (gzip) (a3y17lte PQ3A.190801.002); gzip
content-length:      879
Host:                www.googleapis.com
Connection:          Keep-Alive
Accept-Encoding:     gzip
Protobuf                                                                                                                              [m:auto]
1: eyJhbGciOiJSUzI1NiIsImtpZCI6ImQxMGM4ZjhiMGRjN2Y1NWUyYjM1NDFmMjllNWFjMzc0M2Y3N2NjZWUiLCJ0eXAiOiJKV1QifQ.eyJ1dWlkIjoiYTNhNzM4YmItNWY4My00OWIzLWFiZGItOTUyODA0NDc0ODI3IiwiaX
NzIjoiaHR0cHM6Ly9zZWN1cmV0b2tlbi5nb29nbGUuY29tL3N0YXlzYWZlLXByb2QiLCJhdWQiOiJzdGF5c2FmZS1wcm9kIiwiYXV0aF90aW1lIjoxNjAzOTAxNzY3LCJ1c2VyX2lkIjoiMmExNjJhZGJiMmEyOTUzMmRiYzFjNz
RkYmFhMTIzNzEiLCJzdWIiOiIyYTE2MmFkYmIyYTI5NTMyZGJjMWM3NGRiYWExMjM3MSIsImlhdCI6MTYwMzkwMTc2NywiZXhwIjoxNjAzOTA1MzY3LCJmaXJlYmFzZSI6eyJpZGVudGl0aWVzIjp7fSwic2lnbl9pbl9wcm92aW
RlciI6ImN1c3RvbSJ9fQ.ELov-zOnV6tcPSjoOJN7CnuK4_DaDxaaS9kOfgIGTdH2n5mgGdWMeGZQZgczdEerMIf-5N1LNIwQT5V6mIe1HNPQoaNEn6Jo1zG3kRdMTmub5MIg7YXu9k4Gt9D2pXHWBwQ98tLzpeMxznzklSOTpd3
lUvjfff8kFKaraeYX-E9XDUH_MfupBxOUkRVt4x8QiQG2Wg0oE-nlEC2MFI6wwd0hVkLyaV9rvy3hAhrMTiQnmJyYsi2L3cFu6Wp089ctf5cKfyX80Nc5gU9iw028bijUOkWP3KjdA6dxYLzKBSlVDU9GcrLOtoxr1nik2rx3rvj
F3ZGLTdy1yFiCm3vwmg
```

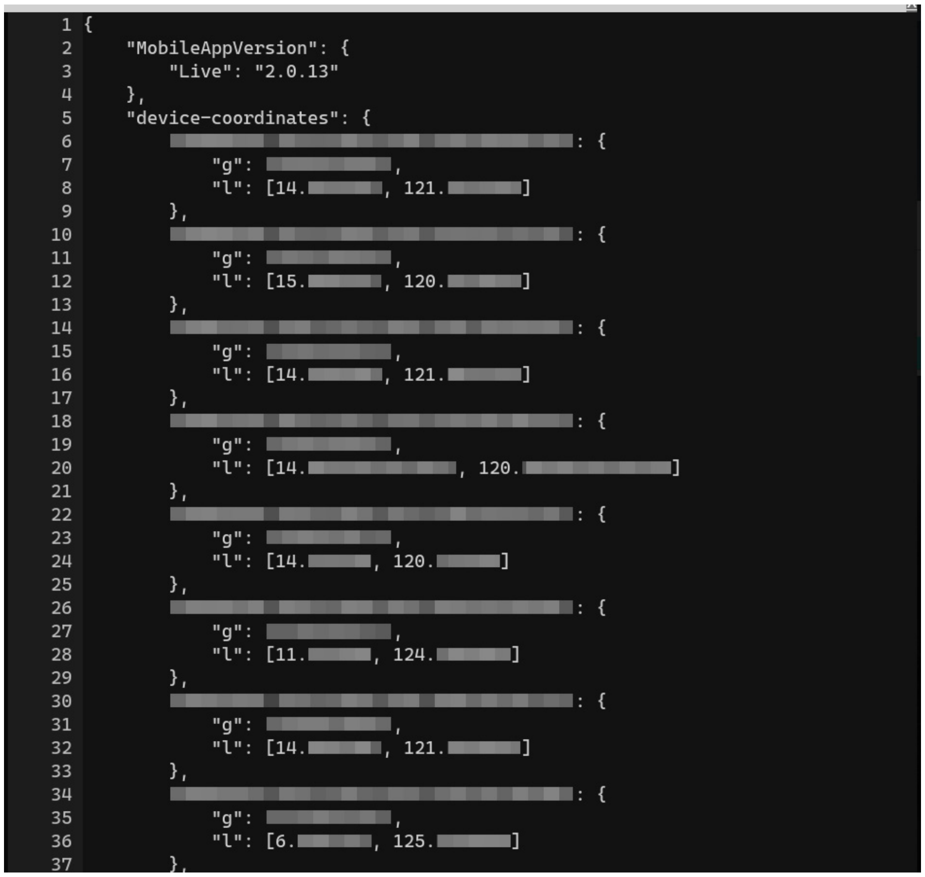
Figure 3: Capture of network traffic from StaySafe PH revealing the Firebase authentication token.

First, we logged in to the app. During this process, we captured a network request to the www.googleapis.com domain (see Figure 3 for an illustration). Second, we recorded the value of the "1" parameter ("eyJhbGci…" in Figure 3) in the protobuf HTTP body, which is an authentication token.

Finally, using the authentication token, we sent the following request to the Firebase Database endpoint to retrieve all records from the database:

```
curl "https://staysafe-prod.firebaseio.com/.
json?auth=$AUTHENTICATION_TOKEN"
```

2 User health information was not stored in this leaked database. Once the app determined nearby users' identifiers (UUIDs), it would query their health information using a separate API at https://ws.staysafe.ph/api/v1/me/trace-registrants



```
{
    "MobileAppVersion": {
        "Live": "2.0.13"
    },
    "device-coordinates": {
        [redacted]: {
            "g": [redacted],
            "l": [14.[redacted], 121.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [15.[redacted], 120.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [14.[redacted], 121.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [14.[redacted], 120.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [14.[redacted], 120.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [11.[redacted], 124.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [14.[redacted], 121.[redacted]]
        },
        [redacted]: {
            "g": [redacted],
            "l": [6.[redacted], 125.[redacted]]
        },
```

Figure 4: User geolocation data (redacted) retrieved from StaySafe PH Firebase database.

In the retrieved data, there were 190,412 items under the key "device-coordinates" (for an excerpt retrieved October 27, 2020, see Figure 4). We sampled a few coordinates and found that they were all within the Philippines. This means that these coordinates belong to app users in the country.

Vulnerability Disclosure

The following table documents our communications to StaySafe PH's developers regarding the disclosure of the issues we identified in this report.

Date	Contact
October 30, 2020	We emailed Multisys, the Philippines Department of Health, and the WHO Philippines regarding the issues we identified with StaySafe PH. We notified them of our intention to publish this research no sooner than 45 days after the date of this disclosure.
November 27, 2020	We received a response from Multisys, stating: "We're currently doing the necessary adjustments to resolve this issue. Our target date to fix this on December 2, 2020 (GMT +8)."

Date	Contact
December 4, 2020	We received a response from Multisys, stating: “We’re still ongoing on the necessary adjustments needed to resolve the issue. It just so happened that some minor issues occurred doing the adjustments. But we’re almost done just need to patch some things up. Will update you as soon as we finish this.”
December 8, 2020	We received a response from Multisys indicating that they had almost completed their fix of the issues we identified.
December 10, 2020	We received a response from Multisys indicating that they had published an app update for Android and iOS.
December 10, 2020	We notified Multisys that the database was still not secured and we could carry out this attack.
December 11, 2020	We received a response from Multisys stating that they had restricted access to the database.

Post-disclosure analysis

On December 10, Multisys notified us that they had released an updated version of the Android and iOS versions of StaySafe PH (version 2.0). Following this notification, we conducted a follow-up analysis and found that, using the same technique as before, we were able to replicate the original issue we identified. We did not test the updated version of the app, as the originally reported technique could still be performed using the previous version (0.12) of the app. This indicated that the security vulnerability still existed in the database backend. Since older versions of the app are easily downloadable, an attacker could, using only public information, learn how to download the geolocation coordinates of all StaySafe PH users. Thus, no fix implemented entirely client-side in the app would be sufficient to fix the vulnerability.

Following this analysis, we notified Multisys that we were still able to replicate the originally reported issue. Multisys responded and notified us that they had restricted access to the database. We attempted to replicate our findings again and found that the same authentication token which previously granted us access to the Firebase database has ceased to do so. In addition, in the previous version of the app (0.12), the “Protect Me” + “Scan Now” interface always showed zero nearby users, suggesting that the feature requiring access to user geolocation data had been removed.

We subsequently analyzed the updated version (2.0) of the app that they released. We found that the “Protect Me“ feature was removed from the interface and that a new “Contact Tracer” feature was added. When we clicked on the “Contact Tracer” button, it showed the message “Account must be tagged as contact tracer". Therefore, since we could no longer replicate the issue using our original technique, we concluded that the issue was resolved. However, there remains the possibility that accounts which are tagged as contact tracers still possess access to other users' geolocation coordinates. If this is true, these contact tracer accounts would become the new weak point. If an attacker compromises a contact tracer account or finds a way to elevate their privileges from normal to contact tracer, they may be able to access the same user data that we had access to using this vulnerability.

Other issues

According to the news organization *Rappler*, Israel Brizuela, CEO of data privacy consulting firm ePrivacyNow and a member of the National Association of Data Protection Officers of the Philippines, alleged that StaySafe PH's "alarming" permissions made it "like borderline spyware." Other media reports, such as by the *Philippine Daily Inquirer* in June 2020, also referenced Brizuela's posts about Philippine contact tracing apps on ePrivacyNow's website (e.g., those published in May and June) when reporting that StaySafe PH required "dangerous permissions," including access to a user's "camera, contacts, location, microphone, phone, text messages, calendar and settings." In addition, Eliseo Rio Jr., a former undersecretary of the Department of Information and Communications Technology (DICT), stated in a Facebook post dated June 12, 2020, that StaySafe PH is "ineffective as a contract tracing app in the Philippines." Lawmakers of the House of Representatives filed House Resolution 1009 in July 2020 seeking an investigation into StaySafe PH, with their resolution citing Eliseo Rio Jr. and Brizuela's posts.

Privacy concerns over the app also led the Inter-Agency Task Force for the Management of Emerging Infectious Diseases (IATF) to issue Resolution No. 45, which orders the Department of Health to sign a Memorandum of Agreement (MOA) with Multisys regarding the "donation" of "the source code, data, data ownership and intellectual property" of StaySafe PH to the Department of Health. Presidential Spokesperson Harry Roque announced in August 2020 that the handover of the StaySafe PH system and its data was complete. However, when *Rappler* asked government officials about the data transfer in December 2020, the news article mentioned that they provided "only vague or contradictory responses." In a Facebook post dated December 4, 2020, Eliseo Rio Jr. also claimed that "[u]ntil now StaySafe has not complied or turned over to the government its aforesaid softwares and data."

A report by Exodus Privacy on the permissions required for StaySafe PH version 0.11.1 of the app indicates that it did not request permission to a user's contacts, microphone, phone state, and system settings. Another report by Exodus Privacy on COVID-KAYA version 1.3.6 shows that the app requested permissions to contacts and phone state, while StaySafe PH version 0.11.1 did not request such permissions. Therefore, it appears that Brizuela's blog posts erroneously combined the permissions requested by COVID-KAYA version 1.3.6 and StaySafe PH version 0.11.1, attributing permissions requested by COVID-KAYA to StaySafe PH.[3]

StaySafe PH provides the option for users to register using a Filipino phone number or with a Facebook account. According to Facebook's documentation, if the user chooses to log in with Facebook, StaySafe PH's developer would receive at least the following information from the user's Facebook account:

- The Facebook account's unique ID
- First and last names
- Profile picture
- Email address

Logging on to the app using a Facebook account allows the developer to associate information collected within the app (e.g., location and COVID-19 status) with the user's online identity (e.g., Facebook account and email address), forming a more complete understanding of an individual's life. At the same time, by logging on using a Facebook account, Facebook would also become aware that the account is using the StaySafe app. Facebook may potentially use this information, for instance, to enhance personalized ads presented to the user.

About COVID-KAYA

The COVID-KAYA app was developed jointly by the Philippines' Department of Health Epidemiology Bureau, the World Health Organization (WHO), and Dure Technologies,[4] a technology company with offices in Switzerland and India, in coordination with the Philippines Department of Information and Communications Technology (DICT).[5] COVID-KAYA is designed for local government units (or *barangays*), hospitals, and laboratories to submit and view COVID-19 cases, contacts, and patient information. The app is intended to be used by health workers and administrators, but not by the general public. An earlier version of the app we analyzed had no "register" functionality for users to sign up independently, but a version that was released on September 30 includes the register functionality.

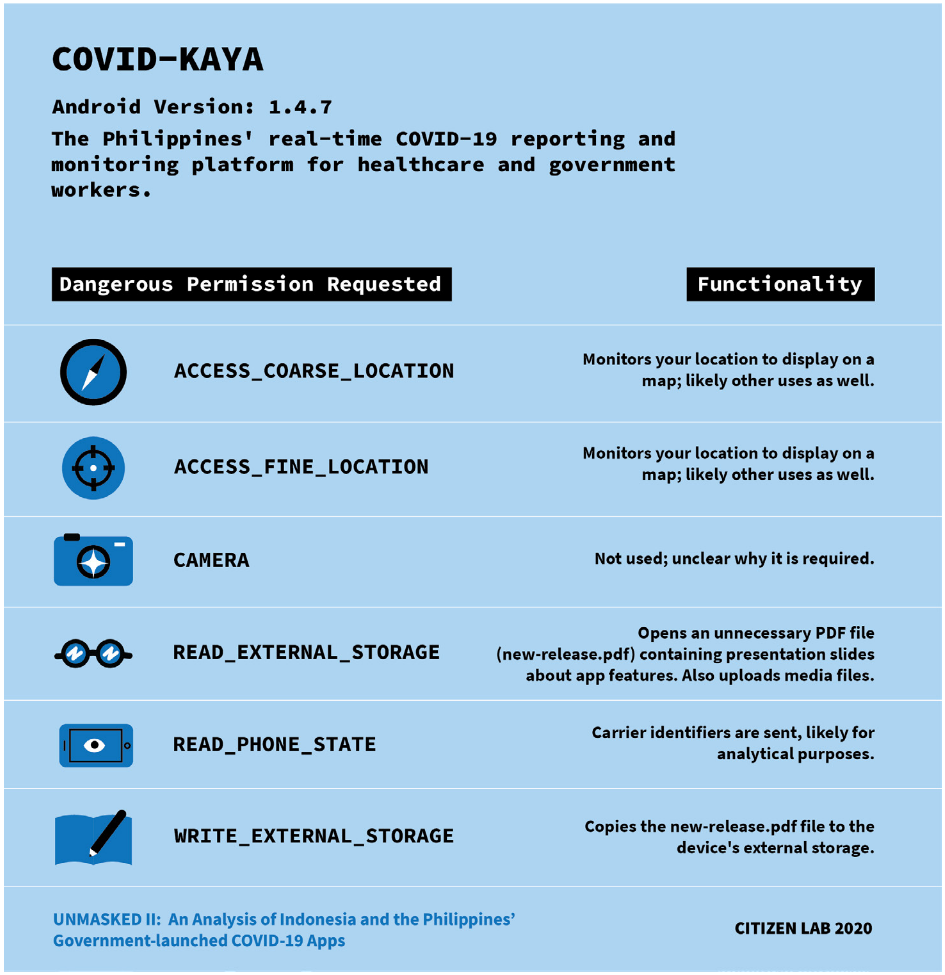
3 In his latest post (dated November 9, 2020), Brizuela again erroneously combined the permissions requested by COVID-KAYA and Stay Safe PH in the same bulleted list.

4 Although there has been no mention of Dure Technologies' involvement with the app in government press releases and media reports, an email address under Dure Technologies' domain (contact@duretechnologies.com) is listed under the Developer section of COVID-KAYA's Google Play Store page.

5 https://pia.gov.ph/news/articles/1043602

COVID-KAYA does not appear to have a dedicated privacy policy. Instead, its Google Play Store page lists a link to the World Health Organization's Privacy Policy. However, the WHO's Privacy Policy only pertains to WHO websites ("all sites within the 'who.int' domain name"), and thus it cannot be applied to the COVID-KAYA app.

Like the other apps we analyze in this report, COVID-KAYA requests dangerous permissions concerning the measurement of a user's geolocation coordinates, use of the camera, and access to shared files. Additionally, we found that COVID-KAYA has access to read the phone's "state," potentially allowing it access to sensitive identifying information (e.g., carrier identifier). Our technical analysis of COVID-KAYA found that it uses many functionalities and permissions that are beyond the app's purposes. However, as we do not have the ability to register an account on this platform, we were not able to test the operation of the app in practice. Thus, the findings we present in this section regarding how COVID-KAYA uses its requested permissions are based largely on analysis of its source code. This type of analysis presents limitations because apps may contain code that is not used in practice.



COVID-KAYA

Android Version: 1.4.7

The Philippines' real-time COVID-19 reporting and monitoring platform for healthcare and government workers.

Dangerous Permission Requested	Functionality
ACCESS_COARSE_LOCATION	Monitors your location to display on a map; likely other uses as well.
ACCESS_FINE_LOCATION	Monitors your location to display on a map; likely other uses as well.
CAMERA	Not used; unclear why it is required.
READ_EXTERNAL_STORAGE	Opens an unnecessary PDF file (new-release.pdf) containing presentation slides about app features. Also uploads media files.
READ_PHONE_STATE	Carrier identifiers are sent, likely for analytical purposes.
WRITE_EXTERNAL_STORAGE	Copies the new-release.pdf file to the device's external storage.

UNMASKED II: An Analysis of Indonesia and the Philippines' Government-launched COVID-19 Apps

CITIZEN LAB 2020

Figure 5: Dangerous permissions requested by COVID-KAYA and their functionality.

Analysis of COVID-KAYA's dangerous permissions

1. Location

We found that the app requests two location permissions, namely ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION.

By reverse engineering the code in the app, we found that these permissions are used to display a map of COVID-19 cases near the user. If the user denies access to these permissions, then the map functionality is unavailable.

The map data was provided by Google Maps, therefore, the user's location had to be sent to Google for it to return nearby map tiles. We are unsure if users' locations were also sent to COVID-KAYA's own servers because we could not test out the app's operation in practice.

2. Camera

Although this permission was requested, we did not find it being used. If this permission is unused, the developers can cease declaring this permission, as it would provide no benefit to the app's functionality.

3. Device storage

The app requests two permissions related to external storage, READ_EXTERNAL STORAGE and WRITE_EXTERNAL STORAGE. By reverse engineering the app, we found that it uses the WRITE_EXTERNAL_STORAGE permission to store a PDF file (named "new-release.pdf") in the external storage. This PDF includes slides showing the app's new features, which is not necessary for the app to function. The contents of this file could be integrated within the app's interface, thus eliminating the need to access external storage. This unnecessary permission would allow an attacker who acquires the ability to remotely execute code in the app to access the phone's external storage.

4. Read phone state and identity

The COVID-KAYA app requests the READ_PHONE_STATE permission. By reverse engineering the app, we found that it uses this permission to collect the carrier name. Such information can be used for user analytics and profiling. READ_PHONE_STATE also allows apps to obtain the phone number(s) associated with the device, although we have not observed COVID-KAYA transmitting such information in practice.

Vulnerabilities discovered in COVID-KAYA

We discovered two security vulnerabilities within the app. The first vulnerability involved an unprotected information retrieval endpoint (API) used internally within the app. This unprotected endpoint can be found easily by slightly changing the URL of COVID-KAYA web app. Once accessed, this endpoint returns a complete list of the app's registered users, including their full names and other personal information.

The second vulnerability we identified is that the COVID-KAYA developer included the app's administrator credentials in its code. If an attacker finds these credentials, they can log into the administrative dashboard, and retrieve the list of the app's registered users, including their full names and other personal information.

We reported these vulnerabilities to the developer on August 18, 2020. On October 29, 2020, we confirmed these issues had been fixed and that the leaked credentials had been invalidated. Further details can be found in our vulnerability report.

Discussion

In the table below, we compare some of the dangerous permissions declared by PeduliLindungi and StaySafe PH apps that we discovered in our analysis against ten countries' official contact tracing apps, as analyzed by Exodus, an automated permission analysis system developed by Exodus Privacy, a French non-profit organization "managed by hacktivists."[6]

Country	App name	ACCESS_ COARSE_ LOCATION	ACCESS_ FINE_ LOCATION	CAMERA	READ_ EXTERNAL_ STORAGE	WRITE_ EXTERNAL_ STORAGE
Indonesia	PeduliLindungi (2.2.2)	✓	✓	✓	✓	✓
The Philippines	StaySafe PH (0.12)	✓	✓	✓	✓	✓
Canada*	COVID Alert (1.0.8)					
Colombia*	CoronApp (1.2.56)	✓	✓			
France*	StopCovid (1.1.4)	✓	✓	✓		
Germany*	Corona-Warn (1.3.1)			✓		
India*	Aarogya Setu (1.4.1)	✓	✓	✓		
Israel*	Hamagen (2.2.6)	✓	✓		✓	✓
Malaysia*	MyTrace (1.0.30)	✓	✓		✓	✓
New Zealand*	NZ COVID Tracer (1.5.0)			✓		
Saudi Arabia*	Tabaud (1.2.0)					
Singapore*	TraceTogether (2.3.8)	✓	✓			

Table 1: *Some of the dangerous permissions required by official COVID-19 contact tracing apps.* Analysis conducted by Exodus.*

6 See https://exodus-privacy.eu.org/en/page/who/

The World Health Organization, as well as several human rights organizations and privacy advocates have published ethical guidelines for the development and use of digital tools to respond to the COVID-19 pandemic.[7] These guidelines share one common principle, namely "data minimization," whereby the app or tool "should collect the least possible information" that is necessary for achieving its objective.

Contact tracing apps that adhere to the data minimization principle do not collect personally identifiable data because contact tracing can be achieved without such information. Protocols such as DP-3T and Apple and Google's Exposure Notification System show that digital contact tracing is possible without collecting sensitive information (e.g., user location data). The official national contact tracing apps of Canada (COVID Alert) and Saudi Arabia (Tabaud) are built using Apple and Google's Exposure Notification System, and, as shown in Table 1, both apps do not require any dangerous permissions.

PeduliLindungi and StaySafe PH's numerous dangerous permissions do not adhere to the data minimization principle, as they collect more data than what is necessary for digital contact tracing. The presence of these permissions is partly due to the functionalities that have been added alongside contact tracing. For example, both apps contain notification functions for alerting users when they are in areas with COVID-19 cases. However, other uses, such as accessing the device's camera to take a profile picture or accessing external storage to store a document describing features of the application, are not only unnecessary, but also inconsistent with the data minimization principle. In the case of StaySafe PH, we demonstrated how the use of dangerous permissions to collect geolocation data gave rise to a vulnerability in which the location data of each user was revealed. Not only did this reveal the locations of StaySafe PH's users, but we suspect that this data could also be used to link real-world identities to in-app identities, thus revealing the health statuses of real-world users as well.

Conclusion

In this report, we analyzed version 2.2.2 of Indonesia's PeduliLindungi app, as well as the Philippines' StaySafe PH app (version 0.12) and COVID-KAYA app (version 1.4.7). Our analysis focused on the Android version of these apps, particularly the "dangerous permissions" required (i.e., "permissions that could potentially affect a user's privacy or the device's normal operation"), such as access to the camera, location, phone status, and other sensitive user information.

7 They include the Electronic Frontier Foundation (EFF), Amnesty International and the Oxford Internet Institute.

Our analysis found that the PeduliLindungi app collects and associates a user's geolocation coordinates with their name, phone number, and WIFI MAC address. We confirmed that the Philippines' StaySafe PH app collects user geolocation data and centrally stores them in an insecure database. Moreover, the COVID-KAYA app requests access to sensitive private information that is not necessary for its principal function, in addition to containing several vulnerabilities. In summary, we found that these COVID-19 apps did not follow the data minimization principle because they request excessive system permission to access and collect information that is unnecessary for the apps' main functions to operate.

In our analysis, we demonstrated that concerns over these apps' failure to achieve data minimization are not theoretical. In StaySafe PH, we show how transmission of geolocation coordinates, which not all COVID-19 contact tracing apps require to function, gave rise to a vulnerability in which the geolocation coordinates of every StaySafe user were downloadable using a single HTTP request. We believe that by observing the coordinates of StaySafe users, these in-app users could be linked to real-world identities, thus also revealing the health statuses of real-world people. In apps that do not collect geolocation coordinates to implement contact tracing, such as ones that use the Google/Apple Exposure Notification System, such a vulnerability would have been precluded from existing in the first place. Our findings underscore the need for developers of contact tracing apps, as well as apps generally, to minimize the data that they collect to help safeguard the security and privacy of their users.