EasyChair Preprint

№ 5046

Covid Tracker Application

Prashant Singh, Ayush Patel and Humaira Hossain

February 26, 2021

Covid-19 Tracker

Prashant Singh
B. Tech SCSE
Galgotias University,

Noida, India
prashant_singh.scsebtech
@galgotiasuniversity.edu.in

Ayush Patel
B. Tech SCSE

Galgotias University
Noida, India
ayushpatel466@gm
ail.com

Humaira
hossain
B.Tech
SCSE

Galgotias University
Bangladesh
humairamethela4@gmail.co
m

Abstract— Coivid-19 tracker is a React based responsive web application which provides you the real time data of current covid-19 cases all across the world. It will show you how many cases are recently recorded and how many recovered as well as the total cases being recorded in a particular country. The app will have a leaflet package of React which will provide us the interactive map thus we will try to make it different and more interactive than other trackers present out there. These maps will enable the user to see real time data of total cases in any given country. The easy to use GUI clubbed with the interactive maps and graphs will help us to provide the vital information that the users need in order to stay updated with the covid-19 situation that is rocking the world and subsequently affecting the economy at a global scale. The web application will be accessible through desktop or mobile to provide the user with ease of access. We aim to provide accurate data. Our aim is to alleviate some stress from the user who feels anxious about being uninformed about what might be happening without them. Our app also serves as a learning opportunity for the members along with a great incentive to help people.

INTRODUCTION

The Covid-19 Tracker is a react based application which gives us the track of current Covid -19 cases and recovered cases as well as the number of deaths recorded. It has worldwide data and country data.The maps will enable the user to see real time data of total cases in any given country.

EASE OF USE:

The web application will be accessible through desktop or mobile to provide the user with ease of access. We aim to provide accurate data. Our aim is to alleviate some stress from the user who feels anxious about being uninformed about what might be happening without them. Our app also serves as a learning opportunity for the members along with a great incentive to help people.

Literature Reviews:

As we already know, in early December 2019, an outbreak of coronavirus disease 2019 (COVID-19), caused by a novel severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), occurred in Wuhan City, Hubei Province, China. On January 30, 2020 the World Health Organization declared the outbreak as a Public Health Emergency of International Concern. As of February 14, 2020, 49,053 laboratory-confirmed and 1,381 deaths have been reported globally. Perceived risk of acquiring disease has led many governments to institute a variety of control measures. We conducted a literature review of publicly available information to summarize knowledge

about the pathogen and the current epidemic. In this literature review, the causative agent, pathogenesis and immune responses, epidemiology, diagnosis, treatment and management of the disease, control and prevention strategies are all reviewed.We are still affected by the disease and facing the problems. It is a global threat and we do have to keep track of it now, and even in the future as well. Scientists all across the world are already trying their best to get a vaccine. It is controlled now in some countries, but not in India. The recovery rate is better than before but that's not enough, the problem is that it still exists. It is necessary to take precautions even if the conditions are better, but doing work is also important, we can't just sit around forever. For that we need a tracker so that we can keep track of cases around us and also stay updated with the data in order to be safe.

II. SYSTEM ARCHITECTURE

The type of architecture adopted for the data collection aspects of tracing apps has been a matter of much discus- sion due to both security and privacy concerns. We will discuss three distinct system architectures commonly used or proposed for developing COVID-19 tracing applications. These are the centralised, the decentralised, and the hybrid approaches that combine features from both the centralised and the decentralised architectures. Our classification criteria consider how the server is used and what data is required (or stored) by it. We now discuss each of the three archi- tectures detailing their salient features. We will discuss some specific tracing apps that employ each of our three architec- tures in a later section.

A. CENTRALISED

Figure 1 shows the main entities and interactions of a cen- tralised architecture. We note that the centralised architec- ture we describe is based on the Bluetrace protocol [13]. The initial requirement for the app is that a user has to pre-register with the central server. The server generates a privacy-preserving Temporary ID (TempID) for each device. This TempID is then encrypted with a secret key (known only to the central server authority) and sent to the device. Devices exchange these TempIDs (in Bluetooth encounter messages) when they come in close contact with each other. Once a user tests positive, they can volunteer to upload all of their stored encounter messages to the central server. The server maps the TempIDs in these messages to individuals to identify at-risk contacts. More details on the centralised architecture's key processes are now given.

1) REGISTRATION PHASE

Figure 2 shows the steps required to register a user in a centralised architecture. A user downloads the app (steps 1 and 2) and registers details such as name, mobile phone number, age bracket, and postcode with the server (step 3). The server verifies the mobile number by sending a One Time Password (OTP) by SMS (steps 4 and 5). Upon verification, the server computes a TempID (step 6), which is only valid for a short time (Bluetrace recommended expiry time is 15 min). The TempID and the expiry time are then transmitted to the user's app.

2) REGISTERING ENCOUNTERS/CONTACTS INFORMATION

Once a user comes in contact with another app user, they exchange an ''Encounter Message'' using Bluetooth, as pre- sented in Figure 3. An encounter message comprises the exchange of TempID, Phone Model, and Transmit Power (TxPower) (steps 1 and 3). Each device also records the Received Signal Strength Indicator (RSSI) and the times- tamp of the message delivery (steps 2 and 4). Note that phone numbers are not included in these messages. Since the TempIDs are generated and encrypted by the server they do not reveal any of the app user's personal information. Thus, both app users have a symmetric record of the encounter that is stored on their respective phones' local storage. The protocol uses a temporary blacklist to avoid a user registering duplicated contacts. Thus, once a user receives an Encounter Message, the app automatically blacklists the sender for a short time.

3) UPLOADING ENCOUNTERS DATA

All encounter records are stored locally and are not auto- matically uploaded to the server. Figure 4 shows the appli- cation flow when a user tests positive for COVID-19 (step 1). The health official confirms whether the user has the trac- ing app installed, and flags the user as infected (step 2).

4) SERVER-SIDE PROCESSING OF THE

UPLOADED DATA

The server iterates through the list of encounter messages, decrypting each TempID with its secret key. This TempID is then mapped to the user's mobile number. The server uses the TxPower and RSSI values to approximate the distance (prox- imity) separating the users during the reported encounter. The proximity estimation can also be performed locally on the phone, but this has battery usage implications. This proximity data, in conjunction with the timestamps, is used to ascertain the risk profile (closeness and duration) of the encounter (step 5, Figure 4). A list is prepared with all the required information (step 6) for further processing by the relevant health official (step 7). To summarise: In the centralised architecture, the cen- tral server plays a key role in performing core functional- ities such as storing encrypted PII information, generating anonymous TempIDs, risk analysis, and notifications for close contacts. This accumulation of responsibilities raises privacy concerns that are discussed in detail in Section III. The server is assumed trusted in this architecture, with some countries introducing strict privacy- protection regula- tions for safeguarding the use and life cycle of the collected data [14].

Problem Formulation:

To begin with the problem, we have to get the current data of COVID-19 cases.

- We have this well-known API which provides the data of well-known diseases one of them is what we need COVID-19.
- Another free API known as "disease.sh" which keeps us updated and provides us with death data, active cases data, total cases and much more.
- API stands for application program interface. A programmer writing an application program can make a request to the Operating System using API (using graphical user interface or command interface). It is a set of routines, protocols and tools for building software and applications. It
- may be any type of system like a web-based system, operating-system or a database System
- We need to fetch the data from that API and use it so we don't need our own database to manually enter the data and keep track of it.
- As we are aware of the fact that data fetched from the API's is in JSON format, so we need to utilize that data and make a better user interface with that.

B. DECENTRALISED

In contrast to the centralised architecture, the decentralised architecture proposes to move core functionalities to the user devices, leaving the server with minimal involvement in the contact tracing process. The idea is to enhance user privacy by generating anonymous identifiers at the user devices (keeping real user identities secret from the other users as well as the server) and processing the exposure notifications on individual devices instead of the centralised server.

We discuss the privacy and security implications of this design in Section III.

We take the Private Automated Contact Tracing proto- col (PACT) [15] as a base to describe the decentralised architecture. The decentralised approach does not require app users to 'pre-register' before use, thus avoiding the stor- age of any PII with the server. Devices generate their random seeds (used as input for a pseudorandom function), which are used in combination with the current time to generate privacy-preserving pseudonyms or 'chirps' with a very short lifetime of about 1 min (see Figure 5). These chirps are subse- quently periodically exchanged with other devices that come in close contact. Once a user is positively diagnosed with COVID-19, they can volunteer to upload their seeds and the relevant time information to a central server. This is in contrast to the centralised architecture where the complete list of encounter messages is uploaded. Uploading of seeds, instead of all used chirps, improves latency and provides improved bandwidth utilisation.

The central server only acts as a rendezvous point, akin to a bulletin board to advertise the seeds of the infected users. This server is considered 'honest- but- curious'. Other app users can download these seeds

to reconstruct the chirps (by using timestamps) that were sent by the infected users. The server, as well as other users, cannot derive any identifying details just by knowing the seeds and chirps. Only the other app users can perform a risk analysis to check if they are exposed for a long enough duration. This one-way lookup against the downloaded seeds restricts the server's functionality and alleviates some of the privacy risks (see Section III). More details on the decentralised architecture's key processes are now given.

1) APP INSTALLATION

COVID-19 tracing apps that adopt the decentralised archi- tecture do not necessarily require an interactive registration process during the app installation stage. The app installation process only

verifies a user's smartphone and deploys a ran- dom seed generation algorithm that is not linked to the phone

2) GENERATING SEEDS, CHIRPS AND EXCHANGING CHIRPS

Once the decentralised tracing app is installed, the seed is generated (with an expiry period of one hour) by the user's device (see Figure 7). This seed and the current time are subsequently used in a pseudorandom function to generate the chirp. The chirps are not linked to an individual or their phone - so in principle, they are anonymous. The app gen- erates new chirps with a time granularity of 1 min. These are broadcasted every few seconds via the Bluetooth beacon. In the listener's phone, the app will automatically store all chirps received (step 4 in Figure 7). The information stored in the receiving app includes the chirp, the timestamp when the chirp is received, and the maximum RSSI value. Identical chirps received within 1 min are ignored. Note the critical difference from the centralised architecture where TempIDs are created by the server - in the decentralised case, the seeds and chirps are generated at the device.

3) UPLOADING ENCOUNTERS DATA

If a user is diagnosed positive, they are given a unique ‘‘permission number’’ by the relevant authority to authorise the upload of all used seeds that are locally stored in their phone (illustrated in Figure 8), as well as the creation and expiry times of the seeds. Note, the server in the decentralised architecture only gets the seeds associated with a single identified user. This is to be compared with the centralised architecture where the complete contact list (with TempIDs) of all encountered individuals is uploaded to the server.

4) THE CONTACT TRACING PROCESS

Contrary to the centralised architecture, the tracing process in the decentralised architecture is performed locally by the app user on their device (instead of the central server). The app users can communicate with the server, typically once per day, to download any seeds uploaded by infected users. Given such seeds are downloaded (step 8 in Figure 8), the user's app then reconstructs all the corresponding chirps (using pseudorandom calculations based on the seeds and discrete-time intervals between the start and expiry time). Finally, the app performs a lookup to check if any of the reconstructed chirp information appears in its local encounter chirp log. If so, proximity and duration times are then derived (based on timestamps and RSSI values) for risk analysis purposes. No human intervention is required.

C. HYBRID

In the centralised architecture, the server performs all the complex tasks, e.g., TempID calculations, encryption, decryption, risk analysis, and notifications of alerts for the at-risk contacts. On the other hand, all these functionalities are delegated to devices in the decentralised architecture, keeping the server only as a bulletin board for lookup pur- poses. The hybrid architecture proposes that these function- alities are split between the server and the devices. More specifically, the TempID generation and management remain decentralised (i.e., handled by devices) to ensure privacy and anonymisation, whilst the risk analysis and notifications should be the responsibility of the centralised server. There are three main reasons for performing the tracing process at the server: i) In the decentralised architecture, the server is unaware of the number of at-risk users as the devices make this risk analysis without taking the server into consideration. Thus, the server does not have any statistical information and is unable to run any data analytics to identify

exposure clusters. ii) Risk analysis and notifications are considered a sensitive process that should be handled by the authorities,

ARCHITECTURE SUMMARY

We have discussed the three base architectures employed for developing applications for contact tracing purposes. The architectures are categorised based on the functionality and level of privacy preservation at the central server. In the centralised architecture, the server manages the security keys, generation of anonymous IDs, contact risk analysis, and notification processes. All these roles are transferred to the devices in the decentralised architecture while the server acts simply as a bulletin board. The hybrid architecture tries to balance the load on the server and improve privacy preservation by splitting functionalities between the end-user device and the server. One distinct advantage of using an architecture that pushes the risk analysis and notification process to the centralised server (i.e., both centralised and hybrid architectures) is that health officials can decide the rate of notifications depending on the pandemic circumstances (e.g., the availability of test kits). On the other hand, decen- tralised and hybrid architectures aim to keep the user identi- ties secret from the central server. A server security breach in this latter architecture would, therefore, result in lower information leakage.

DATA MANAGEMENT, PRIVACY, AND SECURITY

One of the major issues in any tracing application is man- agement, privacy, and security of the data that is collected. The European Data Protection Board issued a statement on the importance of protecting personal data while fighting COVID-19 and flagged articles of the General Data Protec- tion Regulation that provide the legal grounds for processing personal data in the context of epidemics [18]. In addition, some Governments' have passed special privacy protections laws aimed at addressing privacy issues [19]. To comply with these requirements, the tracing apps need to use a multitude of techniques, across the three distinct phases of their operation: i) Registration, ii) Operation, and iii) Positive case identifica- tion phases, depending on:

- What data is produced and by whom?
- What data is exchanged between whom and when?
- What data is stored where and by whom?
- Who can access what piece of data?

In this section, we will first discuss the data life cycle for the three architectures described in Section II and then focus on the privacy and security issues associated with these architectures. We consider three key stakeholders in the attack ecosystem; i) the government ii) the administrator controlling the central server (referred to as server for brevity), and iii) malicious users. All architectures assume that health author- ities3 already know the real identities of all positive cases as all uploads are authorised through the health authorities to prevent fake data being uploaded.

A. DATA MANAGEMENT

Details of data storage in the centralised architectures appear in Table 1. The server is responsible for i) storing PII collected when a user registers, ii) generating, storing and transferring the TempIDs to all registered users periodically (after every 15 min for example), and iii) maintaining a list of all individ- uals who are diagnosed as positive and their close contacts. In contrast, the user device, after receiving the TempID from the server, carries out the following two tasks: i) it generates, exchanges and stores the contacts it has had with peers for a specified period of time, usually 21 days, and ii) upon request, it shares contact data it has stored with the server with the consent of the user.

Data storage details for the decentralised architectures are presented in Table 2. The user devices are responsible for generating the hourly seed and computing the chirps based on the seed and the current time. In addition, they are responsible for exchanging and storing these chirps, the RSSI, and the received timestamp information with peers. There is also the option for the device to store additional metadata, such as location information. The server plays a limited role compared to the centralised architecture. It is only called into action when a user is diagnosed as COVID-19 positive and voluntarily uploads the seeds and time validity data as described in Section

II-B2. This data, stored at the server, can now be used for lookup by other users who have come in contact with the infected user, by reconstructing the chirps using the seeds.

Table 3 shows data storage during the various phases of hybrid tracing architecture. In the operation phase, the devices store all encounters as PET entries in two different tables. The server records the device IDs (with blank metadata such as risk score, notified or not, etc.). The server only obtains the PETs from users who have tested positive and volunteers to upload this information. Another significant difference from the decentralised architecture is that these PETs are not transferred to other devices; rather, other devices upload their PETs from their query table for a risk analysis to be carried out by the server.

B. PRIVACY

The success of any automatic contact tracing app depends on several factors, including: how seamlessly and accurately it can capture close contacts. Another factor is the confidence the users have about their privacy and security when using the app. A naive approach for contact tracing could be to develop a privacy-agnostic system that advertises and exchanges the mobile phone numbers of the participants and periodically registers their location with a centralised server [15]. Such an application would raise serious privacy concerns, and would likely not be accepted by users. Therefore, all the architec- tures have privacy protection built-in. However, the amount of protection provided differs considerably and depends on the attack models, trust assumptions, and the protection measures they adopt [20].

In the previous section, we discussed the data management aspects highlighting the source and storage of different types of data in the three architectures. From a privacy perspective, we classify the data that is to be stored into three categories: i) PII of participants (e.g., names, phone numbers, whether they have tested positive to the virus or not, etc.), ii) Contact advertisement messages (pseudonyms exchanged between devices), and iii) Social/proximity graphs; an indication of the interactions between users and the people they came into close contact with. Each data category has different privacy implications.

We first explore the smartphone's privacy implications, as it is typically less secure than a server. In this case, attacks like theft or coercion (a user being forced or per- suaded) will result in the content stored in the smartphone being revealed. This type of threat is present in all of the architectures. However, the difference between the differ- ent architectures is what is stored on the smartphones (see Tables 1, 2, 3). Data that may be stored on devices, such as details of encounter messages, is considered to be less sensitive, as this information cannot be used to directly iden- tify the contacts.

In a centralised architecture, the servers have access to all three types of data. Therefore, if access to the servers is compromised by malicious users, it would be possible to identify all individuals and their contacts, therefore jeopar- dising their privacy. Hence, centralised architectures need to provide adequate protection of the servers to guarantee user privacy.

In the decentralised architecture, all users can access the public server to download the list of seeds and calculate the chirps used by an infected user. However, as these seeds are uploaded together with their expiry periods, they can result in the unauthorised identification of infected individuals using other side-channel information. For example, malicious per- sons/ apps/ organisations can keep collecting the ephemeral identifiers and the seeds from reported cases and link the identifiers/chirps with the accessed auxiliary information. Also, as only an infected user uploads seeds to the server, a traffic analysis attack, launched by a malicious user who can eavesdrop, would be able to identify a COVID-19 positive user uploading seeds to the server. These attacks are discussed further in section V.

The hybrid architecture adopts additional advanced privacy enhancement methods such as secret sharing [21], decisional Diffie-Hellman (DDH) [22], and private set inter- section [23].4 In general, a user's secret is shared by the user and the server. Furthermore, part of the infection risk analysis is computed at the server using privacy preserving secret shar- ing. Therefore, if one party is

compromised, the entire secret or risk analysis result will not be revealed. These privacy enhancement methods help protect the identity of infected users from being revealed by malicious users or compromised servers. However, these enhanced privacy protections still cannot prevent the users' PII getting de-anonymised if a malicious user can successfully access data collected from side-channel context information.

C. SECURITY

The notion of security encompasses limiting an adversary's abilities to introduce false negatives and false positives in the system, in addition to ensuring system integrity and avail- ability. The motivation for carrying out an attack varies and can range from political and ideological to financial. In the context of contact tracing, an attacker may aim to inject erroneous entries or cause a denial of service.

As all three architectures discussed in this article involve a centralised server, it is pertinent to explore the specific secu- rity threats for each of the architectures. The potential security threat depends on what data originates from a server, what data is shared and accessible to a server and in what form the data is collected and stored (e.g., pseudonymous, encrypted, unencrypted). Furthermore, it depends on the modus operandi

4Readers are encouraged to follow the provided references for details of these privacy preserving, secret sharing techniques.

of the server, namely whether it is i) A trusted server, ii) An honest-but-curious server, iii) A compromised/malicious server, or iv) A colluding server.

A malicious/compromised server can disrupt all types of communications or inject false exposure notifica- tions in all architectures. Similarly, a colluding server can liaise with other malicious entities to perform user de-anonymisation.

In the centralised architecture, the server is considered trusted. It is responsible for storing users' PII and managing security keys used to encrypt/decrypt TempIDs. This poses the risk of data theft if the server gets compromised, a general threat against any centralised server. In this context, the server application needs to run in a trusted environment and use appropriate authentication and access control mechanisms. All information exchanged between the server and the user's smartphone as well as between the server and the health officials needs to be authorised and secure. Thus, centralised architectures only consider malicious users in their attack models and aim to keep the information of all users secure to prevent loss of users' privacy as described in Section V-F. This ensures that no malicious third party can access any information sent/received or exfiltrate information. However, malicious users in centralised architectures could exploit the un- authenticated BLE contact information exchanged between devices to spread incorrect contact information by relaying or replaying. This type of attack, discussed further in Section V-A, would result in false positives during the contact tracing process, forcing users to be incorrectly notified as close contacts.

Decentralised and hybrid architectures, on the other hand, assume an honest-but-curious server that performs all the tasks assigned to it and passively harvests sensitive data, if available. The attack model considers the government and the server to be untrustworthy and only reveals users' identities to the health authorities. As mentioned earlier, the primary user concern relates to the government using the data for purposes other than contact tracing. Therefore, these architectures aim to hide the user identities and generate anonymous IDs for the devices, thereby preventing the ability of the server to link IDs to user information. The decen- tralised architecture delegates data management to users' smartphones, making the solution more robust against a sin- gle point of failure/attack, such as the central server. However, the decentralised architecture still requires a minimally func- tioning central server. Therefore, it will be vulnerable to a much lower number of server-based attacks. In decentralised architectures, anonymous IDs are uploaded to the server, which are then potentially accessible by other smartphones for matching. Thus an honest-but-curious server will not be able to learn any PII, link the anonymous IDs or build social graphs unless it has access to some side-channel information. In case of a data breach, there will be no impact as the attackers only have access to the

seeds/tokens of infected users, which are already public. A malicious user, on the other hand, can still cause false positives by relaying the chirps and launch Denial of Service (DoS) attacks by broadcasting fake but correctly formatted advertisements.

The hybrid architecture carries out the contact risk anal- ysis and notification processes at the server. This prevents any re-identification/ de- anonymisation attacks, as discussed in Section V. In addition, the hybrid architecture provides additional mechanisms to hide user identities from the server while enabling centralised matching of contacts. Similar to the decentralised architectures, it proposes the generation of ephemeral IDs at the devices. The rationale is that devices keep full control over their secret identifiers, making them less susceptible to breaches at the server.

ATTACKS

In this section, we will cover some of the possible attacks that can be launched against different app architectures.

A. REPLAY/RELAY ATTACK

For these attacks, the goal of an adversary is to force the users to store misleading contact information, resulting in false positives. This is achieved by forwarding any message received from honest users at the same or a different location. The adversary requires minimal resources to launch this attack but may use directional antennas to extend the area of its influence further. A relay/ replay attack is the simplest of the attacks that can be launched against users of a tracing app. An adversary can capture the advertised message by a user and immediately relay the captured message at the same location, extending the range of the message, or replay it at another location later on. Note that we classify an attack in this category if the replayed/relayed message has a valid ID (the TempID or chirp); otherwise, it is categorised as a DoS attack (discussed later in Section V-E).

As the TempID has a short expiry time in centralised systems (Bluetrace recommended 15 min), the replay attack can be launched before the expiry of the advertised TempID. If any person who has received this replay message tests positive, the originator will be identified as a close contact of the affected person (false positive) and may be asked to get tested. A more focused attack is also possible if the replay attack is executed near an epidemic testing clinic or a treatment ward/hospital. Individuals already diagnosed with COVID-19 therefore register the replayed messages as a close contact.

The decentralised version has marked differences in behaviour when viewed with the lens of replay/ relay attacks. The chirp generation mechanism, as discussed in Section II-B2, involves using a seed that is valid for 1 hour. The current timestamp randomises the chirps with 1-minute precision. Finally, the receiver records the time at which each chirp is received. During the tracing process, described in Section II-B4, the app validates the received timestamp of each stored chirp with the time of creation of each recon- structed chirp, only accepting the received chirp as valid if these two times approximately match. This mechanism provides safeguards against the replay attack. Theoretically, it can still be launched within 1 minute of the chirp message's expiry time. Relay attacks can still be effective as these are not delayed, resulting in valid chirps.

With hybrid architectures, it is still possible to launch relay attacks, as symmetric information would still exist in the PET tables, maintained by two hosts with a malicious relay. However, the replay attacks are not possible as only one of the users would receive the replayed EphID, and the calcu- lated PETs would only exist for the receiver of the replayed message. If that receiver tests positive, the uploaded replayed PETs would not match with any other PET.

Another difference between the centralised and the decen- tralised architectures w.r.t. the replay attack is the scope

of potential targets. In the centralised version, the victim is the originator (a single person) of the message being replayed while in the decentralised version, victims are the multiple recipients of the replayed message. If the originator tests positive,

he/she will upload the seeds to the server (see Section II-B3). The recipients of the replay messages will identify themselves as close contacts with the originator by comparing the originator's uploaded encounter chirps. On the other hand, in the centralised version, if any person who has received the replay/relay message tests positive, the origina- tor is falsely identified as a close contact. On the other hand, the relay attack has the same purview in all architectures, affecting both the originator and the recipient of the relayed encounter message.

B. WIRELESS DEVICE TRACKING

The attacker's goal in this type of attack is to track the device by the BLE information broadcast by the COVID-19 tracing apps. Consider a shopping mall that wants to track the gen- eral movement pattern of its customers. It can deploy BLE nodes, like Apple's iBeacons, strategically throughout the entire shopping centre, passively listening for advertisements from tracing apps. These nodes can send the captured BLE messages to a central tracking server for further processing. The tracking server can now use simple triangulation [29] and timestamps to estimate the location of each device. This enables tracking, even recording how much time each cus- tomer (device) spends in each store.

For apps that use the centralised architecture, TempIDs and phone model information can be used to to uniquely identify a device. Since TempIDs are changed after a short time (typi- cally 10-15min), tracking a device beyond the point where the device starts advertising a new TempID would require extra intelligence to link the two TempIDs (also see Section V-H) to the same device, advertising the same phone model. In the decentralised architecture, chirps with a 1-minute lifetime provide limited opportunity for tracking. The tracking server can still enumerate the total number of users in the area, how- ever it is difficult to track the movement of a device without a phone model. The tracking, in this case, would be applicable to limited scenarios e.g., a few customers in a shop or if user's device is stationery. Hybrid architectures behave like the centralised architecture as the devices advertise EphID with a lifetime of 15 minutes, making it possible to track a device based on EphIDs.

C. LOCATION CONFIRMATION

In this attack, the attackers' goal is to discover the presence of a user in a known location/environment, such as a neighbour- hood. The BLE advertisements and information contained in the exchange of encounter messages in the centralised archi- tecture can be used to confirm a user's location. For example, assume that Alice is the only one in her family who owns an iPhone 9, and this is known to an adversary, Eve. Eve can con- firm whether Alice is at home by listening to the encounter messages that include Alice's phone model information.

D. ENUMERATION ATTACK

The primary goal of this attack is to count the number of users who have tested positive. Enumeration refers to any user's ability to estimate the number of users infected with COVID-19, who have volunteered to upload their contact tracing data to the server. Note that enumeration does not include the server's ability to count the number of positive cases. In the centralised architecture, the information regard- ing positive cases and their close contacts remains within the server, therefore preventing users from enumerating.

In the decentralised architecture, each positive case uploads all of their seed from the last 21 days (21 days × 24 seeds per day = 504 seeds). All app users can download the list of all seeds from the server and can estimate the num- ber of positive cases. One option to conceal this information is to calculate all the chirps at the server and store these in a bloom filter ([30] and Section VI-B5). This bloom filter (see Figure 13) is then retrieved by the app to check for matches with their contact chirps, without revealing other details. The enumeration attack can also be mitigated, in the decentralised architecture, if the infected user is provided with the capabil- ity to redact some contact information while uploading their contacts. Enumeration attacks are not possible in the hybrid architecture as the server conceals the list of infected user IDs from other users.

FIGURE 13. Encoding chirps into a Bloom Filter.

E. DENIAL OF SERVICE

The goal of this attack is to consume the resources

(battery, bandwidth, processing, etc.) available in the system (user mobile, server). In this regard, we discuss the issue of an adversary injecting bogus encounter messages/chirps into the contact tracing environment. This is done with the following, potentially malafide intentions:

• Consume mobile device storage and battery (all three architectures)

• Cause an upload of these bogus messages to the server once a user tests positive (centralised and hybrid only)

• Increase processing time at the server (centralised and hybrid only)

• Increase processing time at the mobile device (more profound in the decentralised architecture as all chirps (including the bogus ones) need to be compared with the reconstructed chirps)

Note that in the centralised version, the server will process the bogus encounter messages, but will discard these after the server completes a validity check. On the other hand, in the decentralised version, there is no way to check the validity of the received chirp if it is correctly formatted.

F. DE-ANONYMISING THE USERS/ LINKAGE ATTACK

In this attack, a user aims to de-anonymise another user's identity by correlating the anonymous broadcast data with information gathered through side-channels. This can be achieved by linking the anonymous ID with the user's identity in what is known as a linkage attack. Most contact tracing apps have been designed with data and user privacy in mind. However, in the decentralised architecture, it is still possible to identify users once they test positive to the virus. Figure 14 presents the steps required to launch the attack. User A uses a decentralised app that records the details of his encounters with other persons (day/time/duration/ location/gender, etc.) (step 5). If this user receives an alert (step 6), he/she can easily identify the infected user by comparing the reconstructed chirps (step 7). This can be achieved by looking at the time stamp (and duration) of the chirps and comparing his/her collected records (step 8). Some malicious record keeping can also be done automatically by a modified app that collects location information using GPS/WiFi etc.

For the centralised architecture, it is possible to de-anonymise close contacts, but it is hard to de-anonymise a positive case since an app user is not provided with a list of TempIDs for comparison. A positive case can still be identified if a user who is in isolation and has only met one person receives a close contact notification. Tem- pIDs can easily be associated with a user by referring to the advertised mobile model number. The duration of con- tact and an isolated encounter will increase the chances of linking TempIDs with a particular user. Similarly, a Sybil attack [31] can also be launched whereby an attacker can deploy multiple devices and only use a single device for a short time. If the user receives a notification from the server on one of his/her devices, the user can narrow the linkage attack to a short time window when that device was active.

An attacker can launch another kind of linkage attack, called a Paparazzi attack [2], [32] in decentralised apps using passive BLE devices. When a user tests positive, the server receives the seeds, which are, in turn, sent to the users, including the attacker. The attacker reconstructs chirps and combines this data with that obtained from the passive BLE devices. It can then track the positive case throughout the contagion period. Similar to the Paparazzi attack, attackers can trace infected users by deploying a large number of passive BLE devices while colluding with the server. This is referred to as an Orwell attack [33].

FIGURE 15. Summary of apps and protocols.

Required tools:

Major tools required for the development of the applications:

1. **React -** React (also known as React.js or ReactJS) is an open-source JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

2. **HTML -** Hypertext Markup Language, a standardized system for tagging text files to achieve font, color, graphic, and hyperlink effects on World Wide Web pages.

3. **CSS -** CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External style sheets are stored in CSS files.

4. **An IDE (Integrated Development Environment) -** For the complete development and management for the source code and libraries. It's also used for final compilation of the software.

Feasibility Analysis:

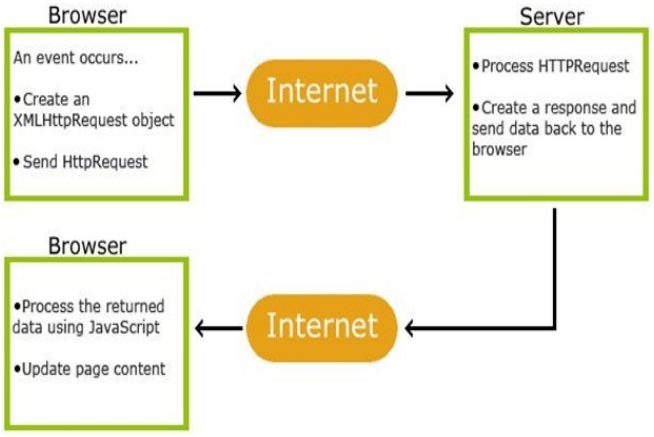
There are a number of trackers present out there, but then what makes our project different from them?

- Simply put you can stay up to date by listening to the news and just looking at the data on some random website. Our app just provides the same with a much better GUI so that wherever you tap on the map you can get the data of that particular country, you can search individually by country name, enable your location to get the data of your country and more.

- Graphs and tooltips are available so that you can just hover your mouse over the
- just simply adding it the home screen will save your time and you can use it as an app in your handsets.

- Our feasibility is attainable as the library used is apt and our goal of providing the users with accurate, and easily accessible data at their fingertips will be achieved.

Figures and Tables:



ACKNOWLEDGMENT

We would like to thank our guide, Mr. Bibhas Kumar for his continual support and unwavering guidance. Furthermore, we would like to thank the numerous moderators for their feedback.

RESULTS

- The application provides an efficient way of showing the identified Covid-19 containment zones to the users in a Google map. With the alarming increase of Covid-19 affected cases throughout the world, this developed application can be employed as a tool for creating further social awareness among the people.
- This application further tracks the user's location and checks whether it is present in the list of identified containment zones. It sends separate notification alerts to the user on entering. Thereby this application identifies the containment zones and highlights the need for taking further precautionary measures for combating Covid-19.
- Tests will be carried out in various containment zones across for the validation of the Android application. The identified containment zones chosen for the testing of the application is yet to be decided.

Reference

https://github.com/disease-sh/api.

"Disease.Sh - Open Disease API (@Diseaseapi) On Twitter". *Twitter.Com*, 2020, https://twitter.com/DiseaseAPI.

"Using Javascript'S Async/Await Syntax To Fetch Data In A React App". *Medium*, 2019, https://medium.com/@matt.readout/using-javascripts-async-await-syntax-to-fetch-data-in-a-react-app-878b930cdc6f#:~:text=Basically%2C%20when%20calling%20fetch(),JSON%20data%20that%20we%20expec.

"How To Make A Chart Using Fetch & REST API's". *Zingchart Blog*, 2017, https://blog.zingchart.com/how-to-make-a-chart-using-fetch-rest-apis/.

P. H. O'Neill, T. Ryan-Mosley, and B. Johnson. (2020). A Flood of Coronavirus Apps are Tracking Us. Now it's Time to Keep Track of Them. [Online]. Available: https://www.technologyreview.com/2020/05/07/1000961/launching-mittr-cov%id-tracing-tracker/

1. P. H. O'Neill, T. Ryan-Mosley, and B. Johnson. (2020). A Flood of Coronavirus Apps are Tracking Us. Now it's Time to Keep Track of Them. [Online]. Available: https://www.technologyreview.com/2020/05/ 07/1000961/launching-mittr-cov%id-tracing- tracker/

2. S. Vaudenay, "Centralized or decentralized? The contact tracing dilemma," IACR Cryptol. ePrint Arch., vol. 2020, p. 531, May 2020. [Online]. Available: https://eprint.iacr.org/ 2020/531
3. C. Criddle and L. Kelion. (2020). Coronavirus Contact-Tracing: World Split Between Two Types of App. [Online]. Available: https://www.bbc.

com/news/technology-52355028
4. J. Duball. (2020). Centralized vs. Decentralized: EU's Contact Trac- ing Privacy Conundrum. [Online]. Available: https://iapp.org/news/a/centralized-vs-decentralized-eus-contact-tracin%g- privacy-conundrum/
5. R. Jennings. (2020).What are the Data Privacy Considerations of Contact Tracing Apps. [Online]. Available: https://ukhumanrightsblog. com/2020/05/01/what-are-the-data-privacy- cons%iderations-of-contact- tracing-apps/
6. D. Palmer. (2020). Security Experts Warn: Don't Let Contact-Tracing App Lead to Surveillance. [Online]. Available: https://www.zdnet.com/article/ security-experts-warn-dont-let-contact-tr%acing- app-lead-to- surveillance/
7. P. Farrell. (2020). Experts Raise Concerns About Security of Coronavirus Tracing App Covidsafe. [Online]. Available: https://www.abc.net.au/ news/2020-05-14/experts-concerned-about-coronavi%rus- tracing- covidsafe-security/12245122
[8] E. M. Redmiles, ''Userconcerns&tradeoffsintechnology-facilitatedcon- tact tracing,'' 2020, arXiv:2004.13219. [Online]. Available: https://arxiv.org/abs/2004.13219
9. J. Li and X. Guo, ''COVID-19 contact-tracing apps: A survey on the global deployment and challenges,'' 2020, arXiv:2005.03599. [Online].
Available: https://arxiv.org/abs/2005.03599
10. L. Reichert, S. Brack, and B. Scheuermann, ''A survey of auto- matic contact tracing approaches,'' Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Tech. Rep. 2020/672, 2020. [Online]. Available: https://eprint.iacr.org/2020/672