What about privacy and reliability, when a Corona Tracing App meets an Android phone?

MaMe82, June 20, 2020

Recently the “Corona Warn App” was released in Germany. After the initial idea of a centralized tracing approach raised massive privacy concerns, it was decided to go with a new decentralized and transparent (Open Source) solution, which relies on Google-Apple “Exposure Notification” API.

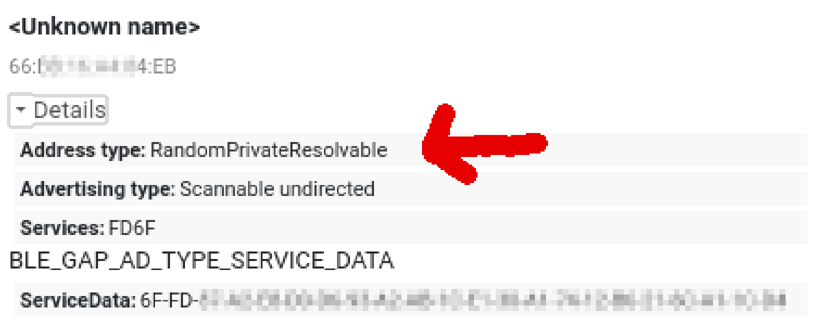
As in my opinion, the idea behind the app is good, I installed it at the day it was made publicly available. Yet, I tried to understand how things work, thus I took a closer look at some things related to Bluetooth Low Energy (BLE) behavior. I have not done any large scale empirical tests, I just inspected my own test device and want to share the results.

The writeup only covers a “Coron Warn App” installation on a Samsung Galaxy S9 running Android 10 (results on my other Android devices are basically the same, no Apple device was harmed, as I do not own one for xxxx reasons).

Also, this write up does not use any meaningful structure. I just list my observations, together with some notes, drawing proper conclusions / verifying my observations is up to the reader (neither do I use formal language, this is no academic paper).

Observation 1 – Advertising with BLE addresses which do not offer highest grade of privacy

One part of Exposure Notification is the BLE advertising of “Rolling Proximity Identifiers” (RPI) and related “Associated Encrypted Metadata” as single payload in an advertising frame using the service UUID 0xFD6F (the brand new “Exposure Notification Service”). I will not go into great detail on the payload parts, as my focus is on how the data is sent.



<Unknown name>
66:[redacted]4:EB
Details
Address type: RandomPrivateResolvable
Advertising type: Scannable undirected
Services: FD6F
BLE_GAP_AD_TYPE_SERVICE_DATA
ServiceData: 6F-FD-[redacted]

The screenshot above shows a captured advertising frame from the “Exposure Notification” service (UUID 0xFD6F). The BLE address used to advertise the data is random, but it is resolvable. Resovable - what does this mean?

To raise my concern, let me very briefly introduce BLE address concept.

Each BLE device has a static address assigned which allows to uniquely identify it. This address is:

- either a "public address" (registered with IEEE, first address part is company assigned , similar to a MAC address)
- or a "random static address" (chosen randomly, not IEEE assigned, but not changeable at run-time/till device power cycles).

Luckily BLE was designed with privacy in mind. The specifications also define "Random Private Addresses", which are meant to hide the device identity (hide the fixed "public address" or "random static address", which always exists). Multiple "Random Private Addresses" could be assigned and changed at run-time.

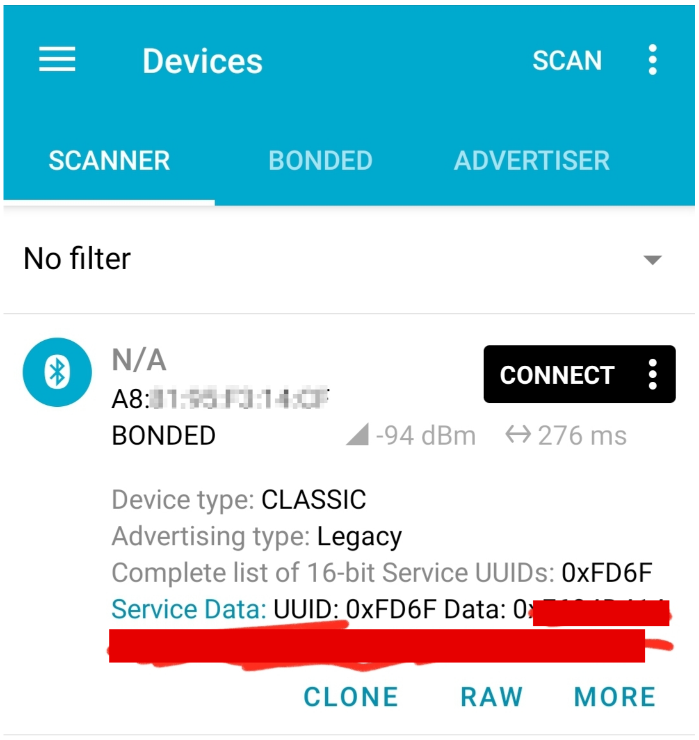
Depending on the BLE version and hardware capabilities "Random Private Addresses" are not always available, that's why not all device are able to run "Exposure Notification" based Corona Tracing apps – they should protect privacy by using random private addresses.

There are two types of "Random Private Addresses":

- Resolvable Random Private Addresses
- Non-Resolvable Random Private Addresses

Randomly chosen private address are a good thing, as somebody monitoring BLE traffic never knows which address belongs to which device (at least not purely based on the address). This is exactly the purpose of those addresses, but it could get an issue if your communication partner changes its private address randomly for each connection. How could you know that you are still communicating with the right party? This issue only exists with "**Non-Resovable** Random Private Addresses". That is why they are only used if nobody should know which device is sending data, not even the intended receiver aka. observer.
Now if two communication parties should be able to identify each other, but third parties - monitoring traffic - should not be able to do so, so called "**Resolvable** Private Random Addresses" are the way to go. A receiver could resolve such an address, if a trust relationship to the sending device exists. To establish this trust relationship, pairing followed by bonding of the two peers is required. During pairing the "Identity Addresses" of both devices are exchange (hopefully in a secure way, as they are based on the "public Address" or the "Random Static Address" which uniquely identifies each device). Along with the identity address, a so called "Identity Resolving Key" (IRK) is exchanged. After pairing those information are usually stored persistently (bonding). If a device wants to use a resolvable address, it generates a random number and, in addition, a hash of this **random number and the IRK**. A part of the hash and the random number are then concatenated to form the resolvable random address. If a trusted peer (which knows the the IRK) observes a transmission from a resolvable address (which could be identified by the two most significant bits) it extracts the random number part of the address, concatenates it with each stored IRK of known and trusted peers and calculates the respective hashes for all of them. If one the resulting hashes matches the hash part of the spotted resolvable address, the sender is known to use the same IRK and thus the stored Identity Address is re-assigned. In result, trusted communication partners could identify each other, while third party observers only could observe random addresses and are not able to resolve real device identities.
For "Exposure Notifications" advertised by Corona Tracing Apps the usage of resolvable addresses obviously is not a good option. There is never a need to connect to a broadcaster of "Exposure Notifications" (it should not even be connectable), nor is there any other reason for being able to resolve the Random Address back to an Identity Address. The usage of resolvable addresses poses an additional risk for privacy. In fact the Exposure Notification Bluetooth Specifications v.1.1 explicitly state "The advertiser address type shall be **Random Non-resolvable**" (https://www.blog.google/documents/62/Exposure_Notification_-_Bluetooth_Specification_v1.1.pdf).

I do not want to construct examples on how an attacker could be able to steal an IRK, in order to resolve the identity address of device which advertises Exposure Notifications. But I want to emphasize something else: The fact that an IRK is exchanged during pairing with a trusted device could lead to the conclusion that a dedicated IRK is generated per trusted peer. That would be difficult, because the broadcaster which wants to use a random resolvable address would have to know upfront which trusted peer is going to connect (in order to generate the address from the correct IRK which was shared with this peer). Instead the same IRK is used for all trusted peers, which means each and every device which was bonded with a smartphone could have received the IRK. This is shown in the following screenshot, where the device which advertises “Exposure Notifications” is marked as “bonded”, because the observer knows the IRK and was able to resolve the random address.



N/A
A8:[redacted]
BONDED -94 dBm ↔ 276 ms

Device type: CLASSIC
Advertising type: Legacy
Complete list of 16-bit Service UUIDs: 0xFD6F
Service Data: UUID: 0xFD6F Data: 0[redacted]

Once more, I do not want to construct scenarios on how to steal pairing/bonding data. If you are interested in this topic, you should be able to find enough publicly available material (f.e. this SySS video: https://www.youtube.com/watch?v=Haaw3iWmzPM)

Observation 2 – Low transmission power, questionable reliability

The “Exposure Notification” powered Corona App estimates a exposure distances to calculate a risk score when required (along with other measurements which are taken into account). In order to estimate the distance to a Bluetooth Low Energy broadcaster device, the “Received Signal Strength Indication” at the receiver’s end is used. The RSSI values alone would not allow to calculate a - more or less - accurate distance (based on signal attenuation), as the information about the signal strength on the sender side has to be taken into account, too. For BLE advertising frames an approximation of transmission power (TX power) used by the broadcaster could be included in the payload, but for “Exposure Notification Service” this is not the case. I had some public discussions on how distance estimation could still work and read available specs back an forth to understand why the TX power information is missing.

In the end I learned that the “Transmit Power Level” data is part of “Associated Encrypted Metadata” which is appended with 4 additional encrypted bytes to the payloads advertised by the “Exposure Notification Service” (https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf)

In the beginning I was a bit confused about this, because some documents stated that received data is disregarded based on calculated attenuation values (“… more than 8 meters (-73 dBm) apart on average...”). I was concerned, that relevant data could be dropped because of wrong measurements if TX power is not taken into account, yet (because it is still encrypted). Luckily, some people which are more involved into the topic clarified that all received encrypted data is stored (at least for 14 days in current configuration). Data is only dropped if considered to be not relevant **after decryption** (at this point all information required to calculate the part of the risk score which is based on signal attenuation are available) or if the 14 days passed by. I do not want to dive into “Exposure Notification Cryptography Specification”, but it should be clear that encrypted data which was captured and stored on a device could only be decrypted, if the corresponding “Diagnosis Keys” (Temporary Exposure Keys) are published (which would happen for obvious reasons).

With some additional explanations from helpful Twitter users, I also learned that sending plain TX power information would open up a new privacy issue, when it comes to user tracking (distance measuring). On the other hand, even with encrypted TX power, the distance could still be calculated once required (diagnosis case, with published encryption keys).

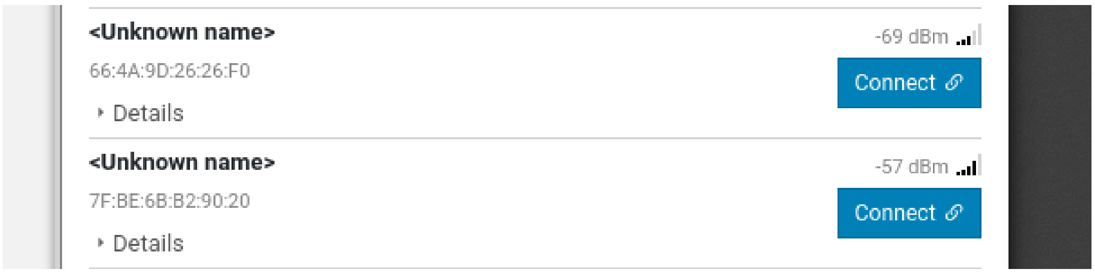
As I learned that TX power is sensitive data, I was curious on how it is actually configured. Is it a fixed value? Is it randomly chosen? If randomly chosen, how often does it change?

I was not able to find any answers to those questions, thus I started to do some RSSI measuring of the “Exposure Notification Service”. To my surprise the TX power seemed to be extraordinarily low. I was not sure if my mobile was broken so I installed an additional application, which allowed me to advertise an iBeacon in addition (with user changeable signal strength, which was set to HIGH).

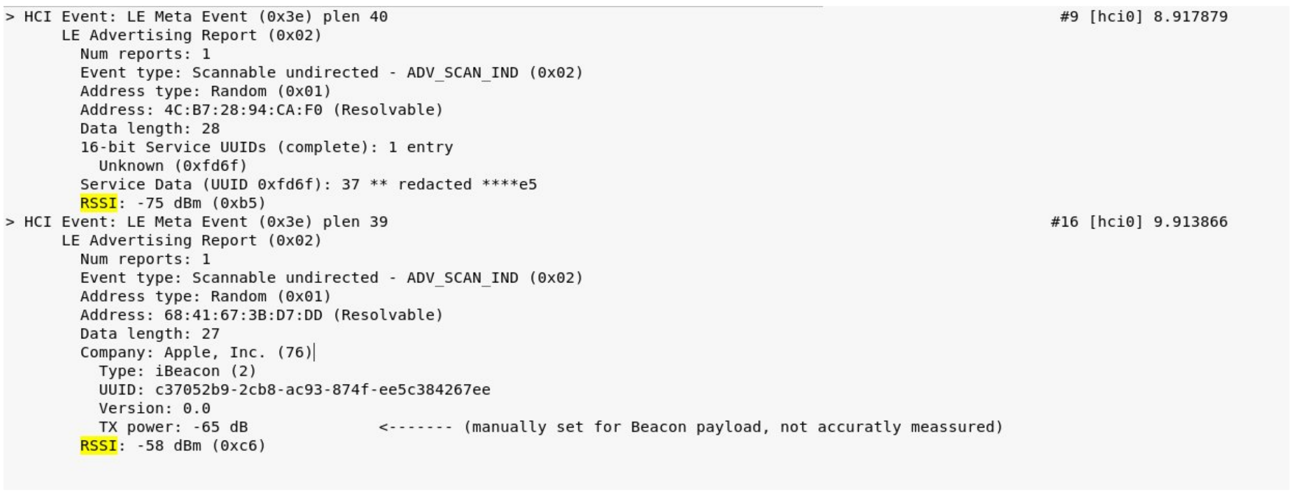
The received signal strength of the “Exposure Notifications Service” was constantly about 10 to 20 dBm below the one from the iBeacon (both send to from the same device). I tested two other Android devices as senders – results were the same.

The screenshots below shows two different measurements taken at a distance of approximately **1 meter** from the device sending exposure notifications.

For the first screenshot a second mobile was used as receiver (high TX power iBeacon at -57 dBm, Exposure Notification from same device at -69 dBm)


<Unknown name> -69 dBm
66:4A:9D:26:26:F0
Connect
Details
<Unknown name> -57 dBm
7F:BE:6B:B2:90:20
Connect
Details

For the second screenshot a notebook with built-in Bluetooth adapter was used as receiver (high TX power iBeacon at -58 dBm, Exposure Notification from same device at -75 dBm)



```
> HCI Event: LE Meta Event (0x3e) plen 40                                                          #9 [hci0] 8.917879
      LE Advertising Report (0x02)
        Num reports: 1
        Event type: Scannable undirected - ADV_SCAN_IND (0x02)
        Address type: Random (0x01)
        Address: 4C:B7:28:94:CA:F0 (Resolvable)
        Data length: 28
        16-bit Service UUIDs (complete): 1 entry
          Unknown (0xfd6f)
        Service Data (UUID 0xfd6f): 37 ** redacted ****e5
        RSSI: -75 dBm (0xb5)
> HCI Event: LE Meta Event (0x3e) plen 39                                                         #16 [hci0] 9.913866
      LE Advertising Report (0x02)
        Num reports: 1
        Event type: Scannable undirected - ADV_SCAN_IND (0x02)
        Address type: Random (0x01)
        Address: 68:41:67:3B:D7:DD (Resolvable)
        Data length: 27
        Company: Apple, Inc. (76)
          Type: iBeacon (2)
          UUID: c37052b9-2cb8-ac93-874f-ee5c384267ee
          Version: 0.0
          TX power: -65 dB              <------- (manually set for Beacon payload, not accuratly meassured)
        RSSI: -58 dBm (0xc6)
```

The received signal strength of Exposure Notification Service was constantly below a tenth of the signal strength of the iBeacon. In order to reach the same low RSSI results as for the the Exposure Notifications, the iBeacon has to sent with a TX power profile of LOW. According to the app used, the LOW setting corresponds to approximately -75 dBm.

I had no equipment around to get more accurate measurements, but it was clear that such a low signal strength means a very short range. Indeed I was not able to receive any signals if I moved the receiver more than 1.5 meters away from the device which was advertising the exposure notifications. According to public discussions there seems to be a different behavior on Apple devices.

For all Android devices owned and tested by myself, this means there is a very high probability, that relevant Exposure Notification can not be received by others, unless I'm closer than 1.5 meters and there are no obstacles between my device and the receiver (the 1.5 meters maximum range have been measured without any obstacles between receiver and device and very few other 2.4GHz activity).

Although I'm aware of the fact, that none of my tests is accurate (no laboratory conditions) they pretty much aligned to an "optimized" real world use case.

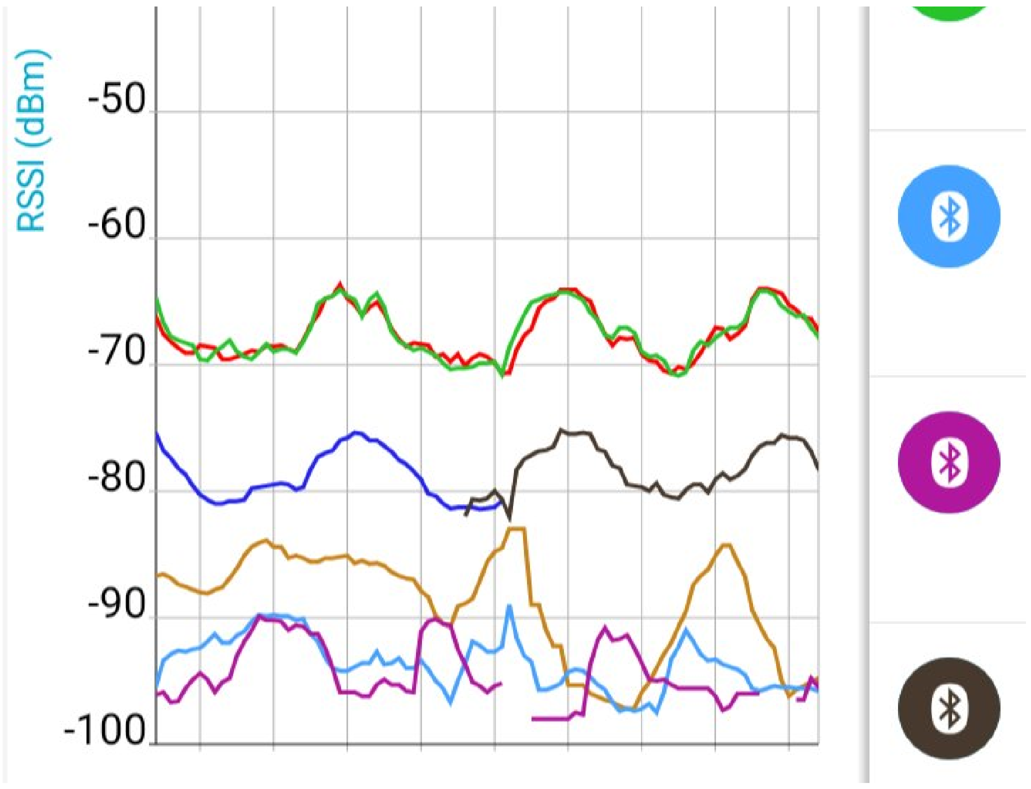
Also, with not being able to receive close range exposure notification (f.e. from 2m distance), taking the calculated signal attenuation into account as part of the exposure risk calculation makes no sense to me – received advertisements are recorded from a distance below 1.5 meters anyways (for greater distances they would not have arrived at all).

Up till now I was not able to make out which part of the application stack is responsible for setting the TX power, the app or the Exposure Notification API (at least I have not invested enough time into making it out).

Observation 3 – If devices advertise additional BLE data chosen privacy measures could get inefficient

As described in the last section the TX power setting of a device, which advertises exposure notifications could be considered as "sensitive data" (because of this, TX power values are encrypted). One of the researchers I talked to, assumed that the TX power settings are changing randomly to prevent accurate distance measuring by third parties.

To find out more about this, I captured RSSI plots of a phone running the “Corona-Warn-App”. The result raised new concerns on privacy. The screenshot below shows such a plot:



The device is running the following three broadcasters:

1. Advertising of manufacturer data (HIGH TX power, **red** plot)
2. Advertising of an iBeacon (HIGH TX power, **green** plot)
3. Advertising “Exposure Notifications” by Corona-Warn-App (TX power supposed to be LOW, **blue** and **black** plots)

The signal strength of the advertising frames from the “Exposure Notification Service” is significantly lower than the signal strength of the advertised iBeacon (green) or manufacturer data (red). Still, the curve progression of the additional advertisings (iBeacon/ maunfacturer data) could easily be aligned to the blue and black plots of the Exposure Notifications (even visually).

This means if additional data is advertised along with exposure notifications, both could be associated to each other without much effort. If this data is tracked along with exposure notifications and related “diagnosis keys” would be published, the additional data could help to deanonymize persons with diagnosis.

Not less concerning: The blue plot ends in the middle of the screenshot, before it continues as black plot. This shows the exact point in time when the “Resolvable Private Random Address” of the device was changed along with the change of the Rolling Proximity Identifiers (RPI). There obviously is no TX power randomization (blue plot continues into black plot, small plot intersection is caused by the processing and visualization of the scanner app).

Ultimately, the RSSI progression of the transmitted exposure notification before the RPI change (blue) could reliably be associated to the new RPIs which are advertised with a new address (black), if additional advertising frames from the same device are taken into account (red/green). This would even work with larger gaps in the RSSI progression, as long as additional data is advertised continuously.

This also indicates that the TX power for advertising exposure notification is set to fixed value.

Summary

The observations described in this document are only on based tests with a limited set of devices owned by myself, some of them are dated (Galaxy S9), some of them are even more dated (f.e. Galaxy S7). But I assume those devices are still used broadly.

Also I assume, most issues arise from hardware limitations, f.e. not being able to assign non-resolvable addresses could maybe be prevented by the Bluetooth hardware in use, ability to correlate TX power settings of different advertisers could be caused by Bluetooth hardware, too.

I would be happy to receive some feedback from users of other devices on my observations. The tests are easy to reproduce.

Stay safe

Cheers, MaMe82