Privacy issues discovered in the BLE implementation of the COVIDSafe Android app

Jim Mussared
jim.mussared@gmail.com
https://twitter.com/jim_mussared
28/04/2020
Last updated: 15/05/2020

Status: Public. Updates ongoing.

Additional notes on BLE MAC randomisation and ID linking

Summary

- The COVIDSafe app has a number of issues which may allow a malicious person to track any user for an indefinite period of time.
- Some of these are unintentional implementation errors, some are deviations from the specification, and some are issues with the specification.
- **Don't Panic!! Users are advised to be aware of these issues but in most cases might reasonably conclude that they are not significant enough to warrant not using the app.**
- If you are concerned, consider turning off Bluetooth except in scenarios where you need the app to be running.
- I still have the app installed (Android) and will continue to do so.

High-level points

- The COVIDSafe Android app contains two implementation flaws that lead to a significant unintended privacy issue, allowing **long-term (many day) tracking of devices**. (Issue #1, Issue #2)
- Long-term tracking of devices has serious implications for tracking people's movements and is a serious invasion of their privacy and a violation of the privacy policy.
- A third, less serious, issue in the design of the protocol leads to a more limited form of long-term tracking, and involves **sharing of data that is not mentioned in the privacy policy**. (Issue #3)
- A fourth issue (new) also in the design of the protocol that in most cases allows for more long-term tracking, and in some cases **exposes the owner's name**. (Issue #4)
- All four issues have been practically demonstrated using cheap off-the-shelf hardware and/or free tools and could be turned into an easy-to-use smartphone app or scanner device requiring no specialist skills to operate.
- A fifth issue allows for **permanent tracking of an iPhone even when the app is uninstalled,** and allows for more cases where Issue 4 exposes the owner's name. It also applies in a different way to Android giving a result similar to Issue #1. (Issue #5) *This issue is not discussed in detail in this document.*
- Recommendations for mitigations are provided for the five issues (and the iPhone background issues, see below).
- It is very likely not possible for the claimed "1.5 metre" distance threshold to be implemented, with **practical testing showing distance cannot be measured accurately enough by this app**.
- **The code can be easily fixed** and a new version of the software released that would completely mitigate these issues while providing no decrease in effectiveness at contact tracing.
    - Further testing of this app is required by people with in-depth BLE experience.

- There is **no evidence of malice, backdoors or any other deliberate violations of user security/privacy.**
- The Singapore team confirm that they have had no direct contact with the Australian team since the initial code drop.
- There is no responsible disclosure program, nor industry best practices like a Bug Bounty. If the code had been open sourced, there would be a clear mechanism for how the community could raise (and even fix) the bugs.
- This analysis was done on version 1.0.11 of the app, released on 26/04/2020.
    - v1.0.15 and v1.0.16 were released 8 days later in the evening of 04/05/2020 but contain no differences in any part of the BLE implementation (only user interface and graphics).

Details were first shared with privacy@health.gov.au at 1:19am on 27/04/2020 with more details later in the day via email and in-app feedback, followed by this document at 5:50pm on 28/04/2020. This document has also been shared with the DTA, ASD, ACSC, and Cyber Security CRC. ~~**No confirmation/communication has been received that these issues are understood, prioritised, or have an estimated time to be fixed/mitigated.**~~ Update: 5pm 05/05/2020, the issues are confirmed.

Update 14/05/2020: v1.0.17 of the Android app was released, containing a fix for Issues #1 and #2, and also can detect and connect to a backgrounded iPhone (by detecting the hashed service uuid). I've also now had confirmation that the remaining issues are being worked on for a future release.

Additional iPhone Summary

- There has been a lot of confusion around whether or not the app works on iPhone when in the background or if the screen is locked.
- Two videos (1, 2) have been released which claim to show evidence that it doesn't work, but it doesn't appear that either actually show what they are claiming. There is lots of media speculation but mostly either without explanation, or reference to these videos. I believe **this is unfortunately causing a lot of iPhone users to choose not to install the app**.
- **Issue #2 in the Android app causes an additional bug in the iPhone app.**
- The analysis specific to iPhone behavior was done on the OpenTrace iOS codebase with the assumption (and observation) that the Australian app is not significantly different.
- It is likely that **the iPhone app can in fact work in the background in many cases** and there are clear fixes to make it work in more cases. (iPhone app running in the background)
- The iPhone app was updated 8 days later on 05/05/2020 but so far appears to not fix these issues.

Author notes

I'm a hybrid hardware and software developer, with current professional experience with open-source development and designing/developing BLE-based products for George Robotics. Formerly worked in programming/electronics education at Grok Learning, and before that at Google Australia as a tech lead in the SRE team as well as some time working with the Android team.

I support the COVIDSafe application and want to see lives saved, but at the same time it's very important to me that these privacy issues are addressed.

Background

The Australian COVIDSafe Android app is based very closely on the OpenTrace app (known as TraceTogether in Singapore) which uses the BlueTrace protocol. OpenTrace was open-sourced after release at https://github.com/opentrace-community/opentrace-android/ . COVIDSafe had not yet been open-sourced.

It helps to understand the BlueTrace whitepaper (https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf) but the most important details are included here.

The main functionality is that the app exchanges TempIDs with other phones nearby. These TempIDs are generated by the server and are encrypted tokens, meaningless to anybody except the server. Only the server holds the decryption key to turn them back into a primary key that is used to look up your name and contact details.

The server gives the app a TempID which it then uses for the next two hours (despite the 15-minute recommendation from BlueTrace). The app will exchange these TempIDs with all nearby phones, recording them in a database on the phone.

If someone tests positive to SARS-COV-2, then they can trigger the app to upload all collected TempIDs to the central server, which can then decrypt them and figure out who to contact. TempIDs are only uploaded to the server if the user clicks the button to do so.

The Android app and the iPhone app work in much the same way, with some additional constraints imposed on the iPhone app for when the app is in the background. This document will be discussing the Android app unless specifically mentioned.

UniqueID lifetime

The lifetime of this ID is important because it will uniquely identify your phone for this time interval. It doesn't say who you are, but it allows any Bluetooth device in range to know that it's the same phone it saw earlier. A simple example of how this could be abused is that it allows someone to track your movement as your ID shows up in multiple locations. Devices

recording these IDs could provide effective distributed location tracing. This is why it's very important that the same ID is used for the shortest possible period.

A slightly more complicated example is that if I already know who someone is and can record their ID (by being within 20 metres of them for a few seconds), then I now can track that exact person. This would be especially worrying to victims of domestic violence (also note that the issues described here do not require access to an unlocked phone).

It may sound abstract to talk about "a device that tracks these IDs", but what we're talking about here is a phone app that would take a few hours for a skilled developer to write (**many people have already written these apps)**, or even a tiny device about the size of a matchbox. And this is not theoretical, I was able to make the latter in about 30 minutes on the Sunday night within hours of the app being released.

So it's very important that these TempIDs do not last for more than two hours. I have identified a mechanism that allows for the TempID to last for an indefinite amount of time, and a second simpler mechanism where a different ID is accessible that identifies the phone in the same way. Both of these issues are due to simple coding errors in the COVIDSafe app that demonstrate that the developers were not extremely familiar with Bluetooth Low Energy, and that the app was **never extensively tested by anyone with the relevant experience to look at the data payloads being exchanged**. This can be done by simple BLE debugging applications available for free in the Android App Store.

An additional note: The Australian app will only obtain a single TempID at a time from the server. This means that if a phone is offline (due to loss of phone signal, or server issues) that **the two-hour expiry will be extended until communication with the server is re-established**. The BlueTrace whitepaper suggests that TempIDs should be downloaded in batches. This is a minor issue but a surprising departure from the design. It is **unclear whether the TempID will even be valid for contact tracing purposes as it contains an expiry time that makes it invalid.**

Technical details

This requires understanding some Bluetooth Low Energy (BLE) basics:

- Devices can advertise their existence via a Generic Access Profile (GAP) advertisement. This generally includes a device name, some metadata about whether they can be connected to by another device, a manufacturer, and what services they support.
    - Phones do not usually advertise, *especially* on a continuous basis. However, the COVIDSafe app enables this permanently and includes the following information:
        - Basic metadata
        - Service ID (for the OpenTrace service)
        - **Manufacturer data** (this is important for the second issue)

- A device can discover an advertisement from another device, and then connect to the device and access the supported services via the Generic ATTribute profile (GATT).
    - The COVIDSafe app does this for any other phones it finds in range.
    - Scanning can be active or passive. Phones typically only scan if they have an app looking for a beacon, and passive scanning is invisible to other devices (they're just listening on the radio).
- GATT Services support a set of "characteristics" which allow BLE to exchange data.
    - The OpenTrace service supports a single service with a single characteristic which allows the two phones to exchange a payload. This **characteristic can be read from and written to**, and the payload includes fields such as:
        - The current TempID (encrypted user ID)
        - **The phone model name** (this is important for the third issue)
- When a phone is a GATT server, the OS also includes other information via other default services.
- When BLE devices communicate they must have a network address (a MAC address).
    - Bluetooth works differently to WiFi and Ethernet which typically only use fixed addresses (generated by the manufacturer).
    - Depending on the scenario, this address can be fixed (for long term pairing of two devices) or randomised (for once-off connections). **COVIDSafe relies on the phone doing the latter** (otherwise this would uniquely identify your device). Address randomisation usually happens on a very short interval (~minutes).
    - **See additional notes at the end of this document** for extra discussion around the MAC address and the potential for another way for the TempID to be tracked long term.

Issue #1

This has been assigned **CVE-2020-12857**

This issue was fixed in v1.0.17 released on 14/05/2020 (see note at end)

The first issue is that the COVIDSafe app caches the characteristic value that will be read by a remote device using the MAC address of the remote device. Due to a logic error, this cache is only cleared when a successful transaction takes place (i.e. the remote device subsequently writes its own payload back to the characteristic). Importantly, the cache entry is not removed when:

- The two-hour expiry finishes for the TempID
- After any sort of short time limit.

This means that a remote device using a static address that never completes the transaction will always see the same cached TempID. This cached value will last until the app is restarted (likely due to a phone reboot).

I have demonstrated that this **cached value can be read back for over 24 hours**, with no limit to how long this could last**.** It is trivial to set up a device with a static address. I was able to confirm that the TempID reported to other nearby phones changed as expected on the two-hour interval, but the old cached one remained the same.

This flaw also exists in the version of the OpenTrace source code that was shared with the Australian team, and has been inherited by the Australian app.

- This line shows the only place the cache entry is removed (when a successful transaction occurs).
- This line shows where the cache entry is created for the remote MAC address.

Recommendations:

- Remove this cache altogether, it's not clear what efficiency gain this cache affords.
- or: Ensure that the cache is updated with the TempID changes.
- or: Set a timeout on all cache entries.
- and: Change the TempID expiry to 15 minutes, as per the BlueTrace recommendation, and download them in batches.

Analysis:

- Cache invalidation is a really common problem in computer science, in fact it's a famously hard problem. However, programmers are trained to look for the ways in which items are removed from caches as part of standard code review and design.
- The transaction not completing is a classic example of dealing with cleanup after error cases.

This issue was first reported to privacy@health.gov.au at 1:19am on 27/04/2020, and subsequently by in-app feedback later that day. It was also reported to asd.assist@defence.gov.au at 4:52pm on 27/04/2020.

This issue was first reported to the Singapore OpenTrace team at 12:38am on 30/04/2020 and was fixed in this commit to the opentrace-android repository at 7:11pm on the same day.

Note on the fix: When this issue was fixed, instead of removing the cache, there was additional logic added to the disconnection event to remove the entry. I'm not aware of any way that this event can fail to fire, but it would have been far more robust and easy to reason about the fix if the cache had been removed altogether.

Issue #2

This has been assigned **CVE-2020-12858**

This issue was fixed in v1.0.17 released on 14/05/2020 (see note at end).

The second issue is much simpler. The advertising payload is generated once at startup and used for the lifetime of the app. Due to requirements by the iPhone app, there are additional

bytes included in this payload added to the manufacturer field. However, **specifically in the Australian app, these bytes are generated once at startup, and then the same data is used continuously for the lifetime of the app**. This means that the advertising payload will consistently identify the same device.

Again, like Issue #1, I have demonstrated that this **advertising payload remains unchanged for over 72 hours**.

Here is an example of the payload with the manufacturer data field highlighted.

`02:01:02:`**`06:ff:ff:03:39:63:31`**`:02:0a:ff:11:07:f9:18:c2:4c:09:fe:f0:80:6a:4f:95:15:fc:b3:2a:b8`

This corresponds to a manufacturer name of "Withings" (0x03ff, see the company identifier registry), with **three extra padding bytes**, resulting in 0xff03**396331**. These three bytes give a 1 in 16 million unique ID. (Update: due to the way the UUID is used to generate this, it's actually "only" 1-in-4096… see analysis below).

In more detail; A BLE advertising payload contains fields in the form:

- 1-byte length (0x06 above means 6 bytes plus the length byte)
- 1-byte type (0xff above means type=255, manufacturer info)
- N bytes of data.

Also note that multi-byte integers are stored in little-endian byte order, which makes them appear reversed in the payload (e.g. the Withings ID of 0x03ff becomes ff:03 in the payload).

The rest of the payload contains standard fields:
`02:01:02` (type=1, advertising flags, general discovery)
`06:ff:ff:03:39:63:31` (type=255, manufacturer data, see above)
`02:0a:ff (type=10, TX power, full)`
`11:07:f9:18:c2:4c:09:fe:f0:80:6a:4f:95:15:fc:b3:2a:b8` (type=7, Service UUID, COVIDSafe service -- b82ab3fc-1595-4f6a-80f0-fe094cc218f9)

The OpenTrace app appears to behave in a similar way. See this function where the random data is generated and then appended to the manufacturer field, however this data is generated every time advertising is restarted (every 180 seconds). **There is a small but crucial difference introduced in the Australian app where the data is then cached and used for the lifetime of the app**. This is evident in the disassembly.

See the iPhone app running in the background section near the end of the document for more information about why these bytes are here and why this has led to confusion about whether the iPhone app works in the background.

Recommendations:

- At the very least, ensure that this payload is re-generated every time advertising is re-started so that the same sequence of bytes are not re-used.

- Better, find a better workaround for the reason the iPhone app requires this and ideally remove this field altogether. Any scheme where identifiable bytes are included in the advertising payload can lead to the ability to link successive BLE MAC addresses. (See Additional notes on BLE MAC randomisation below).
- A more general point about the advertising payload. There should be **no unique data in this payload**. Currently the iPhone client sets the device name field to "TR", whereas the Android client does not. Also there is no guarantee that iPhone vs Android generate the fields of this payload in the same order.

Analysis:

- The re-use of this payload was introduced in the Australian version of the app. It is not present in the OpenTrace code.
- Comments in the open source code could have prevented the Australian developers from misunderstanding the purpose of this code and introducing this modification. It's entirely expected that the Android app and the iPhone app would have been worked on by different people, and so the Australian developers would not necessarily have had a way to realise this implicit link between the two apps. The OpenTrace Kotlin code has almost no comments at all. Programming best practices would encourage commenting code that is as important as this.
- Using a UUID to generate random bytes is very surprising. The first instinct should be something like SecureRandom. In this case however, the UUID is first converted to a string ("UUID.randomUUID().toString()"), which means that each "byte" is actually a hexadecimal char, only taking on 16 possible values 30-39 or 61-66. (Thanks to Chris Culnane for pointing this out).
- Use of the "Withings" manufacturer ID indicates that this code was likely copied from somewhere else without understanding the meaning of the "1023" (0x03ff) value used.
- See the Additional notes on BLE MAC randomisation for why this approach is still flawed and an alternate solution should be investigated for the iPhone app.

This issue was first reported via in-app feedback subsequently by in-app-feedback on 27/04/2020. It was also reported to asd.assist@defence.gov.au at 4:52pm on 27/04/2020.

Note on the fix: The implemented fix was just to undo the regression, the payload is no longer cached.

Issue #3

This has been assigned **CVE-2020-12859.**

The phone model name included in the OpenTrace JSON payload is weakly identifiable. There are enough different models of phones, that it gives a high probability of identifying someone, especially in non-crowded environments. If a house has three residents, each with a different phone, it would allow someone outside the house to trivially identify which of those people were currently at home.

Additionally, the fact that this information is exchanged is **not mentioned in the privacy policy** at https://www.health.gov.au/using-our-websites/privacy/privacy-policy-for-covidsafe-app

The use of this field in the payload is documented and designed into the BlueTrace protocol, so this is not specific to the Australian app. The reason it is included is for calibration of signal strength to physical distance, however it is likely from my own experiments that this calculation does not actually work.

Example values are "Pixel 2", or "Galaxy G8".

Recommendations:

- If you need to compute RSSI->distance so that you can establish which contacts were within a certain threshold, do this on the server instead when the TempID database is uploaded. (This would require the server to also know the phone model of both contacts, but it already knows the name and phone number).
- This needs to be in the Privacy Policy.

Analysis:

- This calibration is extremely dubious but the authors have published details about their methodology. My own research in this area and practical experience suggests that this is highly unlikely to provide useful results. Update: I did some measurements with advertising payloads generated from this app. See also this article.
- There seems to be little benefit in pre-filtering the data on the device.
- This data was deemed by the OpenTrace team to be not identifiable. This is clearly not the case.

This issue was first reported to privacy@health.gov.au at 1:19am on 27/04/2020, and subsequently by in-app feedback later that day. It was also reported to asd.assist@defence.gov.au at 4:52pm on 27/04/2020.

Issue #4

This has been assigned CVE-2020-12860.

A BLE device can be any combination of the four roles: Advertiser, Peripheral, Scanner, Central. Typical combinations are:

- Advertiser-only (a beacon)
- Advertiser+Peripheral (a device like a heart rate monitor)
- Scanner (phone app, looking for beacons)
- Scanner+Central (phone app talking to a device)

COVIDSafe runs in a fairly unique "all of the above" configuration.

When you are a peripheral, you must also be a "connectable advertiser" -- that is, you must advertise, and set the advertisement type to "connectable" -- and then you run a GATT server.

A peripheral must include a set of services containing characteristics and descriptors (the COVIDSafe service is described above). There are several generic services that contain characteristics with useful metadata about the device. One example is 0x1800 "Generic Access" which contains **0x2a00 "Device Name"**. On iPhones, there's an additional service 0x180a "Device Information" which contains **0x2a24 "Model Number String"**.

The device name on Android is set by the user. On most Android phones it appears to default to some variant of the phone model, either exactly, e.g. "Samsung Galaxy S7" or "My new Pixel 2". In this sense it's very similar to Issue #3 but might provide further uniqueness in combination.

However, many people do something like **"Jim's Pixel 2"** as this is the string that shows up in for example a car hands free system -- "Connected to Jim's Pixel 2". I have even seen cases of **"Firstname Lastname's Phone Model"**.

A practical demonstration of this is shown in this video by Richard Nelson. This only takes a few lines of script injected into the COVIDSafe iOS app with an instrumentation tool named Frida -- no modified app or custom scanner app required.

It appears that most iPhones report the value of the device name as just "iPhone" regardless of the user setting. However we have seen cases where the user's setting is returned. (Update: there is a way to trigger this, currently only when the app is foregrounded, which is currently the Government's recommended way to use the app).

The model number string on the iPhone is in the format "iPhone8,2". Similar to Issue #3.

Due to the way Bluetooth works, everything is shared by all apps on the phone. So if multiple apps register services they are all listed together. Additionally, the operating system registers the generic metadata services.

Normally, phones are not peripherals, so unless any app asks to be a Peripheral, there will be no registered services and the device will not be connectable. So although the existence of this data is actually leaked by the OS, **unless COVIDSafe was running, the phone would not be connectable and this data would not be available**. For most people, their phone never would have been connectable during normal operation, and certainly not in background operation.

Note: It's worth being aware of Apple bleee which was published in October 2019 which contains a very detailed analysis of what's available in the additional services registered by an iPhone, due to the AirDrop service, so in many cases this data was already accessible if Airdrop is enabled. However, COVIDSafe forces it for all phones, Android or iPhone.

Many thanks to John Evershed of ProjectComputing, who contacted me from a Facebook post I had written for my friends explaining how the COVIDSafe app works. He noticed in some of his own experiments that the device name he had assigned his iPhone was available when connecting to a peripheral which prompted both of us to look into this in more detail.

Recommendations:

- This is a protocol flaw. The protocol needs to be redesigned to work entirely with advertising payloads only. **There should be no peripheral role or connections involved in the protocol to prevent these additional generic services from being available.**
    - This would fix Issue 5 also, which allows for **permanent tracking of an iPhone, even if the app is uninstalled.**
- During app startup, COVIDSafe should query the system Bluetooth device name and tell the user that this is now being made available.
- **This needs to be in the privacy policy.**
- The Apple/Google protocol is entirely advertiser-only and COVIDSafe should be migrated to that instead.

Analysis:

- This is trivially noticable if you just run a BLE scan using an app like nRF Connect and query all available services and descriptors and look for interesting values. This suggests that very limited (if any) testing was done of this form.
- This is not Android's fault. There is no expectation that a phone should be a full-time connectable peripheral.
- The protocol was clearly not designed with awareness of the default system-provided services that provide this additional information.

This issue was first reported to privacy@health.gov.au, ASD/ACSC, and Cybersecurity CRC at 4:30pm on 02/05/2020.

Issue #5

This has been assigned **CVE-2020-12856**

This is based on collaboration with a researcher and the details are still being finalised.

This affects both Android and iPhone, where in both cases it allows for **permanent long-term tracking of the device, even after the app is uninstalled**.

1. On Android, the execution of this vulnerability is similar to Issues 1-4 above, and is invisible to the target user.
2. On iPhone, there is an additional step required.

Unlike issue #1 & #2, this persists after reboot and factory reset.

This applies to all contact tracing apps based on a similar protocol (e.g. confirmed for UK, Singapore)

I strongly recommend that the Google/Apple protocol is adopted instead. When possible I will add mitigation details.

Lacking an official disclosure/bug bounty program, we have now (5:00pm on 05/05/2020) raised this informally with the DTA and ASD/ACSC though available channels, with as much detail as possible.

iPhone app running in the background

(Caveat, iOS development is not my area of expertise, but I have consulted some other iPhone developers)

Summary

It's not easy to actually show whether the app works or not. It's very easy to make a video that looks like it's showing the app not working, but the app has multiple ways of exchanging TempIDs and you need to conclusively show that all of these mechanisms are not working. Additionally, some proprietary Apple extensions to BLE make detecting this behavior more complicated.

That said, in collaboration with Richard Nelson, we have conclusively shown that there are cases where the app does not work on iPhone, despite claims from the Government that they had fixed these issues compared to the Singapore version.

Updates from Richard are worth reading:
https://medium.com/@wabz/covidsafe-behaviour-on-ios-e4d753fa74ec
https://medium.com/@wabz/the-broken-covidsafe-ios-application-c652d0a462c4

These issues have been reported in this doc and via email, and so have been available to the COVIDSafe team since the morning of Friday 1/5/2020. (And brought to their attention again when reporting Issue #4).

Update: There were some earlier comments in this doc that suggested that there might be an (unconfirmed) issue with rate limiting iPhone connections. This is now confirmed to be **not** true.

First, some details on two of the videos that have been shared widely claiming to demonstrate that the iPhone app does not work correctly in the background. The videos themselves are not terribly important, but the analysis leads to some interesting observations (and bugs) in the way the app works which need to be fixed.

1. First video

The first is https://twitter.com/MikeHaydon/status/1254646784825061377 which shows an iPhone running COVIDSafe, and a second phone running a BLE scanner. The scanner is showing advertising payloads in real time. As the app is backgrounded, you see that the advertising payloads stop appearing. I have spoken to Mike and confirmed this is the behavior he is describing.

The assumption here was that these advertising payloads were in fact the TempIDs being exchanged, but this is not the case. See above for a description of these payloads -- they are only to indicate the existence of a peripheral that another central can connect to. TODO: I have requested a copy of these payloads and will include the description of their contents here.

However, this does not indicate that the COVIDSafe app is no longer functioning, only that it is no longer sending advertising payloads that this particular scanner app can see.

So there are two mechanisms by which the app could still be working:

Firstly, it could still run in Central role running passive scans for a specific service UUID. **This would not show up in a BLE scanner app as scanning is passive**. I have spoken to experienced iPhone developers with significant BLE experience who have confirmed that this is possible, and also seen other posts confirming this, and confirm that the iOS code does scan for the specific service. The code appears to meet all the requirements for background scanning as described in the Apple documentation, including setting the bluetooth-central role. (It's unclear whether this does actually work though due to bugs, see the Two iPhones, both backgrounded? section).

This means that the iPhone should still be able to detect advertising payloads from other devices (iPhones and Android alike), and then initiate the connection and perform the TempID exchange on the GATT characteristic.

Second, while backgrounded, an iPhone can still be an advertiser, but only detectable by other iPhones that are specifically searching for that exact service UUID. The scanner app might not be doing that. Additionally, I have confirmed with a different scanner app (nRF Connect) that a background iPhone is definitely sending advertising payloads using the proprietary Apple extension used in background advertising (example below).

```
02 01 1A
02 0A 0C
0B FF 4C001006131E0EAB71D1
14 FF 4C000101000000000000000000000000000000
```

2. Second video

The second is https://medium.com/@wabz/covidsafe-behaviour-on-ios-e4d753fa74ec which describes an Android phone and two iPhones and includes a video of the Encouter table being populated. I've spoken to Richard about this article, and clarified a few points. However, despite this not at all being his intention, I believe that it would be easy for a reader to reasonably interpret this article as "it doesn't work properly on iPhone". (Edited to add, this post has been updated since this section was written. Additionally Richard has done significantly more analysis and made some much more conclusive results).

There is (was) an "Edit for clarification" paragraph that summarises the article well with two claims.

First claim: "The device with the application in the foreground will log an event for the locked device. The locked device will not initiate a scan, or log anything."

This is hard to understand, because as discussed above, a TempID exchange is symmetric. For a log entry to have been created on one device, it must have been due an exchange of the GATT characteristic. So the only explanation is:

- If the first video above is correct, then the background phone is probably only running in Central role. So it is in fact doing a passive scan, leading to the GATT exchange.
- But somehow the log entry never gets created. (Perhaps a bug in the iOS app? Perhaps writes to CoreData are blocked while backgrounded?)

Second claim: "An Android device will not log events when near an iOS app in the background."

This can be explained by investigating the code for the iOS app. When the iOS app is in the background it will be running in Central mode, passively scanning for the advertisements from the Android phone.

Specifically in Central mode, the iOS code has a cache of Android devices that it has found during scans. It knows they're Android devices because it looks for the extra bytes in the manufacturer field, described in Issue #2. It then uses these bytes as the key to a cache of known Android devices. It then will not attempt to connect to this device for the remainder of this scan interval. This code additionally inadvertently also prevents it from talking to all but one Apple device for a given scan interval.

This wouldn't normally be a problem, because **the Android app is expected to change these bytes** every advertising interval (180 seconds, also the same as the scan interval). However, as described in Issue #2 in the Australian app, these bytes never change for the lifetime of the application.

It appears that this cache is then cleared when the app returns to foreground mode, which will then result in exchanges resuming between the iPhone and the Android device.

So in this video, the cache entry is already populated for this Android device, so no further entries are logged while the iPhone app is backgrounded. (Assuming the scan still keeps running).

Recommendations:

- Fix the Android app as described in Issue #2.
- The saveToCoreData method has no logging on failure (any error is only printed to stdout and then ignored), so it's entirely possible this scenario was missed by the developers. This is a bad coding practice, and should have been spotted in review.
- Rate limit connections to _all_ devices in a way that doesn't require adding identifying information to the advertising payload or allow MAC linking. Ideas:
    - Use the remote device's current random MAC rather than generating other random data. (This might not be possible on iOS).
    - iOS gives you an "identifier" for each peripheral discovered. This is static enough for the duration of the scan that it should be sufficient to implement this cache.
    - Rate limit connections to *all* devices (i.e. only attempt to connect to one new device per N second interval). This may sound like a decrease in effectiveness, however:
        - It's a good idea anyway. Rate limiting connections is a best practice and prevents other unintended consequences like adding too much radio noise.
        - The code inadvertently prevents connecting to more than one iPhone per interval anyway, so it would still be better than the current code.

Analysis:

- The requirement for this cache of Android devices is not explained anywhere. The field is commented in a way that does not explain its actual purpose.
    - Update: I have confirmed with the Singapore team that this is just to rate-limit the communications with Android due to issues they observed.
- The existence of this cache leads to a significant privacy issue for the Android app. (Issue #2) Even if these bytes did change every advertising interval, while it would prevent the very serious long-term tracking issue, it would still be potentially problematic for linking the BLE MAC.
- These videos, **despite best intentions**, are possibly causing a lot of uncertainty about the effectiveness of the BLE app, leading to reduced uptake (my friends and family definitely have fallen for this).
- It's possible that testing would not show the Android issue, as if you just tested that doing a "I have tested positive" action and look at the payload on the server, it would have included all the expected records.
- Better coordination between the iPhone and Android developers would have helped avoid problems.

Two iPhones, both backgrounded?

One of the phones must be an advertiser. It seems like this should be technically possible as advertising is allowed in background mode (also this project), as is acting as a peripheral role. As described above, we've seen advertising payloads that suggest that this is working.

Update: I have confirmed with the Singapore team that background-iPhone to background-iPhone is **not expected to work**. All other scenarios should work, however. That said, the messaging from the Australian Government should have been more clear, as they **have claimed that it should work due to fixes they had made**.

This is not a problem with "older models" or "antenna issues" as claimed.

There are code issues in the Singapore OpenTrace code that explain why backgrounded mode doesn't work correctly. It's not clear from the behaviour or disassembly that any changes have been made to these three bugs in the Australian version. Please see this document for more detail on the first two bugs preventing scanning from working in the background past the first interval. Thanks to Richard Nelson (@wabzqem) for all the work here investigating this behavior. He has demonstrated with his own test apps that it is possible to make an app scan in the background and communicate with COVIDSafe on other background iPhones or Android phones.

The main suggestion he has is that **in background mode, "stopScan" should never be called**, as there will never be an opportunity to start the scan again until foregrounded. If at least one iPhone is able to scan and connect to peripherals (including backgrounded iPhones), then most of the cases should be covered.

Summary:

- The Central role code is clearly buggy and it's not at all clear if it's actually expected to work.
- Multiple sources indicate that fixing this should be possible, including our own test apps.
- This code wasn't at all tested, otherwise the debug log statements would have shown what was happening (assuming it's actually expected to work).
- The map of known Android devices is preventing Android devices from being repeatedly encountered in a scan interval.
- While backgrounded, the scan interval never restarts.

Additional notes:

I have contributed some minor details of these issues in the document maintained by Geoffrey Huntley and some members of the linked Discord channel are likely aware of the issues.
Geoff's disassembly of the Android app was very useful in investigating these issues.

The "nRF Connect" app by Nordic Semi, available on the Android App store, was also invaluable.

More detailed analysis of the tracing and tracking issues is provided in a post by researchers Chris Culnane, Eleanor McMurtry, Robert Merkel and Vanessa Teague. They make a **very strong recommendation that the TempID expiry should be reduced to 15 minutes**. The team has also provided a great deal of assistance to me while putting this document together.

Additional notes on BLE MAC randomisation and ID linking

It's not clear that the BLE MAC randomisation is synchronised with the TempID rollover. This means that there is the potential for a brief interval where the same device MAC will report the old and new TempID on two subsequent connections. This allows an automated scanner to link TempIDs for the same device together over multiple two-hour windows.

The reverse is also true, all random MAC addresses for the lifetime of the TempID can now be associated. This is an argument for making the TempID interval even shorter (~minutes). So together this means that a malicious device can build a history of all TempIDs and MAC addresses for all COVIDSafe devices in range.

To put that more simply: A TempID can link two successive MACs, and a MAC can link two successive TempIDs, resulting in a complete chain of both spanning an indefinite time period.

Unless the TempID and the BLE MAC are changed on exactly the same interval, there will always be a way to link them. This has implications for the identification of other Bluetooth traffic originated from that device.

I have not created a practical demonstration of this, but my brief analysis of the OpenTrace open source code (and the decompiled COVIDSafe code) shows that this has not been taken into account. However, this is just my preliminary analysis and involves some much more complicated code than the snippets highlighted above. Additionally, Kotlin/Java on Android is not my area of interest/expertise, but it would be very easy for the right people to verify this.

Ensuring that these intervals are synchronised ino order to avoid this issue is a feature that is explicitly called out in the design from the Apple/Google collaboration.

In addition to TempID / MAC linking, ***any* two identifiers that update at different intervals will result in linking**. One interesting case to consider is if the COVIDSafe app ever migrates to the Apple/Google protocol. The app can't just suddenly change protocol from one version to the next, as that means that there will be a time period where some amount of users can't talk to the remaining users. So a dual-model app might be required for the

transition. **However, the two protocols running concurrently could lead to the ability to link IDs**.

(I was made aware of this by a friend who works in computer security who suggested I look into how the COVIDSafe app handles this issue)