Security analysis of the NHS COVID-19 App

MAY 19, 2020

☐ Reading time ~33 minutes

Security analysis of the NHS COVID-19 App

Dr Chris Culnane, chris@culnane.org,
Twitter: @chrisculnane,
Independent Security and Privacy Consultant,
Honorary Fellow University of Melbourne,
Visiting Lecturer University of Surrey

Vanessa Teague, vanessa [at] thinkingcybersecurity.com
CEO, Thinking Cybersecurity Pty. Ltd.,
A/Prof (Adj.), Australian National University.

19 May 2020

The following security analysis was conducted via a static analysis of the released source code for the UK's COVID-19 Contact tracing Android app and an evaluation of high-level design documents.

An earlier version of this report was shared with the NCSC on 12 May 2020. We would like to thank NCSC for their rapid response to our report and the constructive dialogue that has taken place since. We have refined the document to clarify a number of points and, where applicable,

include a broad summary of their responses. A full response is available at: https://www.ncsc.gov.uk/blog-post/nhs-covid-19-app-security-two-weeks-on

This post is cross-posted at StateOfIt.com and https://github.com/vteague/contactTracing.

Contents

Introduction

The UK's COVID-19 tracing App contains well-designed protections against many of the attacks that threaten a contact tracing scheme. Although we are not convinced that the perceived benefits of centralised tracing outweigh its risks, we acknowledge that the cryptographic protocol of the UK's app includes a much better effort at mitigation of most external attacks than, for example, its Singaporean/Australian counterpart. The generation of ephemeral encrypted IDs by the app, rather than on the server, significantly improves both privacy and integrity against a

malicious or compromised server. At least, it should, after the main patches described in this document are complete.

We also appreciate that both the app code and a detailed whitepaper have been made available before large-scale deployment. This is a great benefit for exactly the kind of detailed analysis and improvement we now suggest. We refer to Whitepaper Version 0.1, 3rd May 2020 and (Android) app beta code as accessed from GitHub on 7th May 2020.

No cryptographic protocol can meet its security goals if its assumptions are not met. Secure protocols can also be undermined when composed of different components with inconsistent assumptions, particularly in cases where encrypted and unencrypted data are transmitted side-by-side. In this report we show the following.

- In the presence of an untrusted TLS server, the registration process does not properly guarantee either the integrity of the authority public key or the privacy of the shared secrets established at registration. The result completely undermines core security goals of the protocol, including its privacy and its resistance to spoofing and manipulation.
- In the presence of an untrusted TLS server, the storing and transmitting of unencrypted interaction logs facilitates the recovery of InstallationIDs without requiring access to the Authority Private Key.
- Long lived BroadcastValues undermine BLE specified privacy protections and could reveal additional lifestyle attributes about a user who submits their data.
- The monitoring of interactions at 8 second intervals could create unique interaction signatures that could be used to pairwise match device interactions, and when combined with unencrypted submission, allow the recovery of InstallationID from BroadcastValue without access to the Authority Private Key.
- The use of a deterministic counter to trigger KeepAlive updates risks creating an identifier that could be used to link BroadcastValues over multiple days.

The white paper refers only briefly to the establishment of shared secrets at registration time, mentioning in a footnote that ``It would be better if both client and server contributed to the entropy in the [InstallationID]. This may be changed in the future.'' We show here that this is not a nice-to-have for the future, but a critically important foundation for core security goals of the protocol. Fortunately, we believe it is fairly easily achieved by reusing the mechanism already in use for sending symmetrically encrypted IDs.

Protocol summary

The implemented protocol is very similar to what has been described in the NHS App Security Paper, so we will not repeat the details and only provide a high-level overview.

The system adopts a centralised approach in which devices register with a small amount of information (partial postcode) and are in turn issued with a random InstallationID. This ID remains static throughout the operation of the App, and is referred to in the code as the sonarId.

The authority's public key is distributed, along with the InstallationID and a symmetric key for performing the HMAC, during registration. We will refer to the latter as the 'HMAC key.' Although the authority's public key is referred to as the 'server public key' in the code and white paper, it is important to note that the corresponding private key does not need to be stored on the registration or data collection servers - it is the private key of the NHS, and is needed only for the decryption of contact events for exposure notification. We therefore refer to the public key as the 'Authority Public Key,' so that it is not confused with the server's TLS public key. TLS is used to perform the HTTP request, however, there is no indication that certificate pinning is being used, possibly because the service is run through CloudFlare as a Content Delivery Network.

Once every 24 hours the App generates a new ephemeral Elliptic Curve Key Pair. This key pair is used with the Authority Public Key to perform an offline ECDH key exchange. From that exchange an AES symmetric key is generated that is used to encrypt the dates, InstallationID, and CountryCode. We will refer to this as the 'AES key' to distinguish it from the symmetric key used for HMACs. The encryption algorithm also produces an authentication tag. This collection of data is called the BroadcastValue.

Somewhat confusingly, the BroadcastValue is only part of what is broadcast. The full broadcast payload is:

P = (countryCode + BroadcastValue + txPower + transmissionTimeBytes + hmacSignature)

The HMAC Signature is constructed using the HMAC key issued to the device during registration, and covers the following values: countryCode, BroadcastValue, txPowerLevel and transmissionTimeBytes.

It is important to note that the transmission time, Country Code and power level are all broadcast unencrypted. We shall revisit the use of the HMAC in the Section on weaknesses and mitigations against a corrupt or proxied server.

As noted in the NCSC description, once connected the devices exchange range data every 8 seconds. Log data is stored in clear text on the device, protected only by the inbuilt App separation provided by Android/iOS. As such, someone who has root access to the device will be able to read the full log.

If a user has symptoms and decides to upload their log data, the record of observed BroadcastValues, the timings of the contact, the list of RSSI values from the 8 second measurements of distance during the contact, and the InstallationID are packaged for upload. The NCSC white paper claims the following: “The log is integrity protected by an HMAC, encrypted with the shared symmetric key established at registration and sent to the service infrastructure over TLS-protected channels.”

However, we can find no evidence that the data being uploaded is encrypted with the shared symmetric key. Perhaps that sentence was intended to say “The log is integrity protected by an HMAC, *computed* with the shared symmetric key established at registration…” It is clearly HMAC’d but appears to be submitted unencrypted, beyond TLS. The implications are described in the section on Unencrypted log uploads

Attacker model

The importance of distinguishing between an untrusted server and an untrusted authority

The white paper’s technical analysis assumes, “In our model, the infrastructure provider and the healthcare service can be assumed to be the same entity.” However, for many people deciding whether to use the app, trusting the NHS might be quite different from trusting its infrastructure providers, which include US companies such as Google and Cloudflare.

It could be argued that since the server is controlled by the Authority then it is as secure as the Authority Private Key. However, that is a false equivalence. Secure systems are designed to minimise the quantity of material that must be kept absolutely secret. In the current system, the entire user database (including InstallationIDs and HMAC keys) must be kept as secret as the Authority Private Key, which is going to be a difficult task if there are millions of users and records. The number of people with access to the server, either through maintenance or support roles, is vastly larger than the number who should have access to the Authority Private Key. Furthermore, the server is publicly facing, potentially receiving requests from millions of users. Its risk profile is considerably worse than that of the private key.

The security of the communications from the public facing server are also not solely a product of the authority’s actions. TLS Proxying is common place, in which TLS connections are terminated prior to eventual end-point and inspected for security and monitoring reasons. This approach is deployed within the system, with CloudFlare acting as a TLS Proxy and Content Delivery Network in front of the actual server.

The whitepaper is not entirely clear on which service performs the decryption of logs - we assume it is probably the risk modelling service, and that more detail will be available when the server code is made open. The important point here is that, if the registration process is upgraded as we describe below, then only the service responsible for decryption will need to know the Authority Private Key *and no other services will need to hold cryptographic secrets* including the HMAC key or InstallationID.

As such, we distinguish between an untrusted server and an untrusted authority. We assume the authority will adequately protect its private key, holding it securely offline with strict access control. Conversely, we do not assume the server is immune from breach or unauthorised access either internally or externally. As such, we assume the following about the current configuration:

- The authority will adequately protect the private key and it will therefore *not* be accessible to an attacker
- The authority will make best efforts to protect the contents of the server, but cannot guarantee prevention of unauthorised access
- The authority will deploy TLS to protect communication, but is not able to guarantee the security of the TLS Proxy

With the above in mind we proceed to examine what impact an attacker could have were they to compromise the server or the communication between the server and the end-user.

Core security properties not achieved against a malicious TLS proxy.

The whitepaper describes several core security goals. The most relevant to this report are:

> *4) It should not be possible for an external observer to associate any Bluetooth transmission with any device-specific information*.
>
> *5) It should not be possible to submit spoofed data on behalf of another user.*

We take (4) to include identification of the logged Bluetooth transmissions that are uploaded when a person self-diagnoses with COVID-19.

We also suggest two other important goals, which are not explicitly mentioned in the whitepaper but seem important.

> *9?) It should not be possible to alter the event log that a user uploads when they self-diagnose with COVID-19*.

> *10?) It should not be possible to silently prevent a user from successfully operating the app*

> *and submitting their contacts should they wish to*.

The HMAC computed over the event log is clearly intended to achieve Goal 9. However, like Goals 4 and 5, it fails against a compromised or malicious server that knows the HMAC key. We show below that these three goals can be achieved, and attacks against Goal 10 somewhat mitigated, by improving the registration process. We are not able to guarantee Goal 10 against an actively malicious TLS proxy.

Weaknesses and mitigations against a compromised or proxied server.

Distribution of Public Key during Registration

The problem

The Authority Public Key is downloaded from the server at registration time, without any certificate checking.

The implications

This introduces the risk that the initial communication could be intercepted by the TLS Proxy (CloudFlare).

If the transmissions were subject to alteration, and the interception capability was persistent it would be possible to replace the Server Public Key, intercept any uploads, decrypt them, and then forward them on or drop them entirely.

This would also mean that a user's BLE messages, when logged by other users, could not be properly decrypted by the NHS, thus preventing the person from being notified if one of their contacts tested positive for COVID-19. This breaks Goal 10.

Correcting the problem

The Authority Public Key is not secret and does not change frequently. It should be distributed with the app rather than distributed during registration. If it seems important to update the Authority Public Key without an app update, then distribute a Certificate Authority Public Key with the app and distribute the Authority Public Key at registration with a certificate that is carefully verified.

NCSC Response

NCSC clarified that the intermediate certificate pinning was in use, which limits attack to the contracted TLS Proxy (CloudFlare). The suggestion to independently verify the the Authority Public Key, either through inclusion in the App itself or via a Certificate Chain, remains advantageous as it mitigates the risk of compromised TLS Proxy, even if such a compromise is considered to be a low risk event.

Distribution of InstallationID and symmetric HMAC key during Registration

The problem

The InstallationID and HMAC key are downloaded from the server at registration time.

The implications

A compromised or malicious TLS proxy learns the secrets that are supposed to be shared between the user and the NHS. There are several possible attacks.

1. *The attacker can use the HMAC key to identify the user from their Bluetooth transmissions, either when they are first broadcast or when they are uploaded because a contact tested positive.* This would work as follows: whenever the attacker sees a Bluetooth transmission, it runs through its dictionary of known HMAC keys, testing each one to see whether the last 16 bytes of the HMAC it generates on the payload match the HMAC Signature of the transmission. This allows it to identify the InstallationID by retrieval from its dictionary, though it cannot (and doesn't need to) decrypt the payload. This breaks Goal 4.

2. *An attacker with knowledge of InstallationID and the HMAC key can obviously generate spoofed broadcasts that are indistinguishable from the user's true broadcasts.* This in turn would facilitate the creation of fake contact events. This breaks Goal 5.

3. *If the attacker with knowledge of the HMAC key can intercept the infected user's upload of their event logs, then it can drop events and recompute a correct HMAC.* If it also knows the InstallationID and HMAC key of other users, it can insert forged contact events from those users into the upload. This breaks Goal 9.

Correcting the problem

Unlike the public key, the HMAC and InstallationID cannot be shipped with the app, because they are supposed to be different for each user. To address passive attacks from a compromised TLS

Proxy, and assuming that the public key is securely shipped as described above, it is easy to reuse the techniques for generating AES symmetric keys to generate Installation IDs and HMAC keys as well. At registration time, the app could generate a new Elliptic Curve Key Pair. This key pair could be used with the Authority Public Key to perform an offline ECDH key exchange. From that exchange the InstallationID and HMAC key could be derived.

This exactly mimics the code already used to generate ephemeral AES keys, but is used to generate persistent shared secrets instead. This has the great virtue that the HMAC key is never stored in plaintext on the server, and indeed does not need to be derived explicitly until the back-end is trying to decrypt a contact's BLE broadcasts.

This does not entirely eliminate trust in the server-side TLS Proxy (CloudFlare), since it could still intercept and generate an entirely fake initial registration, causing valid submissions to appear to invalid, and facilitate the generation of fake BLE events, however, it does resolve passive attacks. Legislative protection should have been enacted to explicitly protect the data in question and it should be made clear during the consent process that the data will be sent over CloudFlare.

NCSC Response

NCSC have confirmed that there were existing plans to refine the registration process.

Problematic Design Decisions

The issues described below would also be exploitable by a compromised proxy server, however, we distinguish them on the basis that rectifying the proxy server issue does not mitigate the fundamental problems with the design decisions. In particular, the design creates a situation in which the uploaded data needs to be secured to the same level as the Authority Private Key, which as we have already described is an unrealistic expectation. This creates two further attack models, one in which the server is compromised, and the other in which unauthorised access is granted, or data shared with a broader group of people.

In the case of an external attacker compromising the server, the attacker model is similar to a compromised proxy server, except they would gain access to the full data set, not just what they had observed. This will make some of the problems described below more effective in terms of scale of attack.

In the case of unauthorised or broader access, the concern is that there is an incorrect perception of the security of the uploaded data. This could result in a broader group of people having access to the server or the data than would be permitted for the decrypted data or the Authority Private Key. As we will show, the problematic design decisions result in multiple opportunities to recover

the InstallationID from the encrypted value, without the need for access to the Authority Private Key. In essence, if someone would not be granted access to the decrypted data or the Authority Private Key, they should also not be granted access to the uploaded data. As such, the attack model includes both the malicious adversary described above, and internal misclassification of the security of the data.

Long lived BroadcastValues (Encrypted IDs)

The problem

BroadcastValues have a lifetime of 24 hours which facilitates device tracking over a period of 24 hours, undermining the privacy protections native to BLE, as well as revealing some lifestyle attributes.

The implications

BLE is designed to have MAC address randomisation, with a recommended cycle time of 15 minutes in the Bluetooth Core Specification. This is to prevent scanners, of which there are many, from tracking a device over a period exceeding 15 minutes. However, as has already been discussed in regards to the Australian Contact Tracing App, if constant values are broadcast via BLE that exceed that 15 minute window, then the in-built protection becomes redundant. In the case of the UK, the period is extended to 24 hours, which is considerably longer than even the 2 hours of the Australian app. (Though we have no reason to believe that the UK app suffers from the same failures to update at the advertised frequency that the Australian App does.) Note also that the privacy advantages of inbuilt BLE rotation could be undermined even by a 15-minute rotation if that rotation was not synchronised with it. This is one of the advantages of the Apple-Google API, quite independent of the preference for centralisation.

The justification for such a long period is based on evaluating social distancing, not contact tracing. That is not the primary purpose of the app, nor consistent with what the public believe the app to be doing, nor sufficient justification for compromising the privacy of users and facilitating widespread device location tracking. The privacy risks associated with such a long period are also not adequately expressed to the end-user.

Furthermore, when someone self-diagnoses and uploads their logs, access to just the encrypted BroadcastValues that they have received risks revealing a number of lifestyle attributes about the uploader. For example, by comparing the BroadcastValues recorded on the device between 3am and 5am, and subsequently between 11pm and midnight, the viewer will be able to determine whether the uploader woke up and went to bed with the same person, or more revealingly, if they

did not. Further examination of occurrence of BroadcastValues during the day will allow inference of whether the person is in a relationship with someone from work, or whether they potentially met someone after work. The NCSC wrongly dismisses the concerns with social graphs and re-identification, incorrectly assuming the establishing someone's name is a prerequisite for re-identification, whereas it actually occurs whenever additional information, beyond what was anticipated, is learnt about an individual. Re-identification of social network graphs without demographic information has been demonstrated, for example by Narayanan et al 2011.

Correcting the problem

The lifetime of BroadcastValues should be reduced to less than 15 minutes to ensure they do not undermine the intended privacy protections of BLE.

NCSC Response

As already conveyed in public documents, the length of the BroadcastValues is under review.

Monitoring of the interaction every 8 seconds creates a unique interaction signature

The problem

Monitoring active connections with 8 second pings creates an interaction signature that may facilitate pairwise identification in upload data, permitting the recovery of InstallationIDs without needing access to the Authority Private Key.

The implications

If two users upload their data it may be possible from just the record of the 8 second pings to pairwise match their interactions. In doing so the InstallationID would be recovered without needing access to the Authority Private Key.

If two users Alice and Bob, who have met each other at some point during the last few days, both upload their contact records to the server, it should not be possible to discern from the upload that they had met, without first decrypting the BroadcastValue and recovering the InstallationID (Goal 4). It should also not be possible to link either Alice's or Bob's InstallationID (sent during the upload unencrypted except with TLS) with their respective encrypted BroadcastValues. However, due to the detail created by recording the RSSI every 8 seconds, which is uploaded unencrypted, an attacker who is able to observe the uploads or gain access to the server, may be able to learn which users have interacted and thus the mapping of InstallationID to encrypted BroadcastValue

without needing access to the Authority Private Key.

An interaction between Alice and Bob will be defined by the start time, the end time, and the proportional changes in RSSI between those times. The start and end time will be the same for both devices, and that alone may be enough to uniquely identify them. As we showed in previous work, timing alone may act as an identifier Myki Re-Identification. Even if it is not sufficient, when combined with the record of RSSI values a unique signature will be created. The record of RSSI values will form a discrete time-series of an interaction. If we cross-correlate those time-series between devices, the devices that interacted should have a higher correlation than those that did not. In other words, both devices should exhibit an increase in RSSI as the devices get closer, and a decrease as they get further apart, at the same points in time. In effect both devices are recording the same changes in distance, and as such, they are creating a shared signature of the interaction.

It may seem that such interaction signatures will not be unique, but there is significant entropy in the timing and RSSI values associated with an interaction, certainly sufficient to uniquely identify pairs.

Due to the InstallationID being submitted with the log data as well, once a pair is established it allows the mutual linking of plaintext InstallationID to encrypted BroadcastValue by examining the corresponding received BroadcastValues on each device in the pair. This can subsequently be leveraged to link the device to further interactions on the same day, including those which are more fleeting, due to now having the mapping of InstallationID to BroadcastValue.

Correcting the problem

The detailed interaction data should be treated as equally identifying as the InstallationID and encrypted accordingly. Further justification for the necessity and frequency of the pings is required.

NCSC Response

NCSC assert that additional pinging is required to accurately model interactions. They have confirmed that the contact modelling work will be published when completed. Encryption of the logs and uploads is planned, which will resolve the issue in terms of potential for linking prior to decryption.

keepAliveCharacteristic value is deterministic

The problem

keepAliveCharacteristic (8 second pings) uses a predictable incrementing counter to trigger notifications.

The implications

In certain circumstances it might be possible to link successive BroadcastValues across two days by interrogating the KeepAlive counter. This would allow device tracking beyond the already excessive 24 hour period.

In order to record the distance between two devices every 8 seconds during a contact, the app updates the value of the keepAliveCharacteristic. Having done so it issues a notification to the other device, which has registered for notifications of changes in the value of the keepAliveCharacteristic. It is therefore necessary for the underlying value of the keepAliveCharacteristic to regularly change, although the actual value is unimportant - it is the notification message triggered by its change that is of interest in measuring the RSSI.

To implement the required regular change in value there is a class level byte counter, keepAliveValue, that is incremented once every 8 seconds. This counter has a maximum range of values between -128 and +127. As such, the counter will overflow after 34 minutes of connected time, where connected time is time when at least one other device is connected. Despite the overflow, the value of the counter can be predicted, provided there is an assumption that in the intervening period at least one other device has always been connected. This results in the potential to link an observable device across two days if the observer is either in its presence during the changeover (midnight) or within a period in which the device would always have had at least one other device connected to it.

The counter is not externally synchronised, so different devices will have different counters depending on when they first received connections and how many minutes they have been connected since the app was started. This results in the keepAlive value being able to act as a subset identifier for a range of devices that are observable. For example, if two devices, A and B are connected to the attacker, C, at midnight the BroadcastValue will be recreated and it should not be possible for C to link the new BroadcastValues with the old. In reality this might occur because the changeover is not synced with the MAC address cycling, but assuming that it is, it can still potentially be linked via the keepAlive counter. If device A has a count of 24 and device B has a count of 50 at 23:59:00. Immediately after midnight, when the BroadcastValue has been updated, device C can distinguish them from each other, knowing that device A should now have a count of 32 and device B should now have a count of 58, given the next keepAlive message after midnight will be 8 increments from 23:59:00. Even if device C is not observing A and B at midnight, provided both have remained connected to at least one other device their values can be predicted

hours into the future. The only limit is the likelihood of a device remaining connected.

Correcting the problem

The requirement is only that the value of the characteristic is changed in order to trigger the notification. As such, the values do not need to be incrementing. If they were selected at random - ensuring a different value to the current is selected - then there would be no way to predict the future values and it would no longer act as an identifier.

NCSC Response

NCSC confirmed the vulnerability described and have committed to resolving it with a high priority.

Unencrypted log uploads

The problem

As described in the Protocol Summary Section, when an infected person's logs are uploaded to the server, their integrity is protected by an HMAC, however, they do not seem to be encrypted.

The implications

Submitting the data unencrypted allows a number of the attacks described above to be performed by anyone able to observe submissions, i.e. the a TLS Proxy or an attacker who is able to access or compromise the upload server. If the InstallationID - HMAC key list of users are stored on the same server, then a wholesale InstallationID recovery will be possible by undertaking the same HMAC dictionary attack. Even if that is not the case, the pairwise matching of timing data, or RSSI values, will facilitate recovery of InstallationIDs between pairs in the uploaded dataset. Furthermore, the leaking of lifestyle attributes described above will also be possible, all without access to the Authority Private Key.

Correcting the problem

The uploaded data should be encrypted using the same technique as is used for the BroadcastValue. It is important to note that the claimed encryption, that does not appear to be present in the app, was with the shared symmetric key (the HMAC key). Even if this were present it would not be sufficient, because the shared secret may have been compromised. As such, it should use the more sophisticated ECDH key generation approach instead. Likewise, this would not have been a problem had the data been encrypted with the Authority Public Key when stored

on the device in the first place. As such, correctly addressing the lack of protection of local storage, detailed below, will resolve this problem as well. Encrypting the data does not preclude sending the HMAC because being able to identify that an upload came from a particular user is not the same as being able to identify individual BroadcastValues for a user. As such the HMAC on the upload can still be retained to prevent injection of malicious content.

NCSC Response

NCSC confirmed there was a plan to implement further protection on uploads, which will follow in a future app update.

Inadequate protection of local log files

Data stored unencrypted on the device

The problem

Data stored on device is not encrypted, beyond the inherent BroadcastValue encryption. This allows anyone with access to a device to utilise the data for surveillance.

The implications

Whilst the data is protected against some unauthorised access via the built-in app separation security, it is not protected against root level access, from either a party with control over the device, or law enforcement. This presents a number of problems: 1. Surveillance of individuals who are subject to domestic control or abuse; 2. Law enforcement access to detailed surveillance records of interactions.

Regarding point 1., the local log will provide evidence of interactions, including their length and closeness. Whilst they will not reveal identities, by analysing the timing information, it will be possible to extend monitoring beyond a control period. For example, an adversary could establish the set of BroadcastValues for known associates, by observing their initial interaction and matching the timestamps. Any interactions beyond that set will be discernible and could place the victim at risk. In effect it provides an easy tool for adversaries to assert their control beyond periods of direct observation.

One example of such would be if an abusive partner wants to monitor their spouse's interactions. At the very least they will be able to interrogate their spouse about the details of every interaction. In a worse case, if they suspect the spouse of meeting with someone they do not

want, they need only get within range of that person briefly on the same day to record the BroadcastValue themselves and subsequently cross-reference it with those recorded on their spouse's device.

Regarding point 2., if law enforcement gain access to multiple device logs, they will be able to use the methods described above, either timing and RSSI analysis, or HMAC dictionary attacks, to determine interactions between devices. Putting aside the legitimacy of such access, the contact tracing app itself should not result in increases in potential surveillance.

Correcting the problem

The log data should be encrypted with the Authority Public Key when on the device itself, and as already recommended, the detailed 8 second interaction data should be treated as sensitive and be equally protected.

NCSC Response

NCSC confirmed there was a plan to encrypt log file data on the device, and that this will be part of a future app update.

Source Code Access and Responsible Disclosure

The problem

Whilst it is admirable to make the app code available before wide deployment, the GitHub repository includes a disclosure policy we consider it to be incompatible with the principles of responsible disclosure.

The implications

In particular, the clause that provides unilateral control to NHSX over publication of vulnerabilities, perhaps indefinitely, is wholly unacceptable within a responsible disclosure policy, and is not a clause we would be willing to agree to. Security researchers - and the NHS - have an ethical responsibility to make problems known to the people affected by them. The only genuine controversy is how much time should be allowed for correcting them before they are made fully and honestly known to the public.

The solution

Adopt a more traditional responsible disclosure policy with a 30 day period, or for a more

nuanced approach refer to the Hackerone guidelines https://www.hackerone.com/disclosure-guidelines

NCSC Response

NCSC confirmed this had been rectified within a few days of notification, the policy document on GitHub has been adjusted and they now refer to the Hackerone guidelines.

Legal, Commercial and Political Issues

Distinct from the technical analysis above, we also believe there are some broader issues to be considered. We acknowledge that these may fall outside of the technical remit of NHSX or NCSC. They are included here as contributions to the broader discussion that should be taking place in advance of the roll-out of the app. As such, these are opinions, not technical analyses.

There has been significant public discussion in Australia over whether the US CLOUD Act would compel Amazon, which hosts Australia's central server, to share the information with the US government. In the Australian protocol, the AWS server's complete visibility of all relevant information is not easily avoided.

We are not aware of any similar discussion in the UK, though the role of Cloudflare (and perhaps Google) seems just as fundamental. However, there is a crucial difference between the two countries' situations: the modifications we describe in this report would allow the separation of the authority public key (to be held by the UK NHS presumably) from the TLS server's information. This would mean that Cloudflare (or whoever compromised Cloudflare), or other TLS proxies (or whoever compromised them) would receive registration data and the metadata associated with each confirmed infection, but lose the ability to identify the user based on their BLE messages, or forge BLE messages that passed HMAC verification.

More broadly it is our opinion that the data associated with contact tracing should be protected by legislation from use by law enforcement, or any usage not directly related to COVID-19 prevention. There should be a legal requirement that at the end of the crisis all data collected by the app is securely deleted, and not just "anonymised" or repurposed. Australia has begun the process of providing some of these protections, and whilst we would argue there are still a number of critical gaps, it is considerably more than the non-existent protections in the UK. The app should not be a backdoor for data collection for any purpose other than helping address the current crisis.

Furthermore, requiring the public to enable Bluetooth on their devices will have an impact on their privacy overall, enabling commercial profiling and tracking as a side-effect. It is

understandable that compromises must be made at this time, but suitable legislative protection should have been provided to ensure the public do not suffer a loss of privacy as a side-effect of installing the Contact Tracing app. In particular, there should be an absolute ban on use of any application data for purposes other than contact tracing. Australia has drafted some legislation towards some of these goals, although a number of gaps remain Australia COVIDSafe Exposure Draft. So far the UK has not asserted that similar protections will be forthcoming. Any such legislation should prevent the usage of Bluetooth for profiling or tracking throughout the crisis period to best protect the privacy of users and encourage sign up and utilisation of any Bluetooth Contact Tracing app.

Conclusions

The primary advantage of decentralised exposure notification, such as the Google/Apple API, is privacy of the contact graph against the central authority - that is, the government, NHS, or its contractors and service providers (or whoever gains unauthorised access to their databases). The decentralised model does reveal more about users to each other (though the Google/Apple API contains some mitigations for this). A full examination of the trade-offs is beyond the scope of this report - there are attacks on both integrity and privacy in both models: https://eprint.iacr.org/2020/399, but for us the advantage of contact-data privacy against a centralised authority is a sufficiently strong advantage to favour the decentralised approach. For others, we feel that this trade-off should be made more explicit in the UK's public discussion of the centralised model, so that the relative merits of the different models can be better understood. It is also important to recognise that there are different security and privacy properties associated with the different implementations within each model. It is for this reason that it is so important to discuss the specifics of a particular implementation, and not just abstract models. The UK has assisted in that regard by making its code open source, and publishing technical details, before the national roll out. It is vital that process continues, in particular where changes to the fundamental approaches in the app are considered. Updates that alter those privacy or security properties should not be rolled out without sufficient time for consideration and consultation.

There are admirable parts of the implementation and once the already mentioned changes and updates are made, many of the concerns raised in this report will have been addressed. However, there remains some concern as to how privacy and utility are being balanced. The long lived BroadcastValues, and detailed interaction records, remain a concern. Whilst we understand that more detailed records may be desirable for the epidemiological models, it must be balanced with privacy and trust if sufficient adoption of the app is to take place. As such, it may have been beneficial to evaluate what data could be accurately collected by the technology, at sufficient population scale - considering the privacy and trust issues - and from that build the best

epidemiological model. Otherwise there is a risk of placing the cart before the horse, and building a great model that will receive insufficient data, which is not going to help.

Obtaining sufficient scale will require building trust and protecting privacy. The open availability of the source code, and the NCSC's positive interactions with security researchers, will go a long way towards this. However, the messaging around the app, and in particular suggestions of broadening the data collected, combined with insufficient legislative protections, a lack of siloing of the data, and no sunsetting of the data retention or usage, risk undermining the trust that has been earned.

SECURITY COVID-19 PRIVACY CONTACT TRACING

UPDATED ON MAY 20, 2020 BY CHRIS CULNANE

Read More

Internet Voting - From bad idea to poor execution

Presentation at Kawaiicon 2019 - Wellington New Zealand Continue reading

An Update on the Assistance and Access Bill in 2019
Published on February 12, 2019

Assistance and Access Bill 2018
Published on August 30, 2018