HelloTalk Precise GPS Location Disclosure via Unencrypted Local Database (CVE-2020-25900)

Isopach · June 5, 2026

Cve Mobile Privacy

An old finding from 2019 that I never managed to get a public advisory for. I requested a CVE in late 2020, MITRE only approved CVE-2020-25900 in August 2023, and it then sat in RESERVED state because I never published a reference. I was busy with life and didn't get around to writing it, so writing it up now as the public reference so the record can finally be published.

CVE-2020-25900

Affected Versions

HelloTalk for Android, com.hellotalk versions up to and including 3.4.1.

CVSS Base Score: 6.5	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Fixed Version

The local database file under `htbackup/` was encrypted in a later release rolled out around August 2019. The vendor never published an advisory or release-note entry tied to this issue, so there is no exact fixed-version number to point at; the issue was no longer reproducible by the time I revisited it in 2020.

Introduction

HelloTalk is a popular language-exchange app for Android and iOS that pairs you with native speakers of a language you are learning. At time of report it had tens of millions of users worldwide and a heavy concentration of users in East Asia. A common use case is to set your country or city on your profile so other learners can find language partners near them, with the expectation that *only* the country or city is shared, not your home address.

In March 2019 I was poking around the on-device storage of the Android client and noticed that the app kept a local backup directory called `htbackup/` containing an unencrypted SQLite database. Inside that database was a table that stored full-precision GPS coordinates of every user whose profile I had ever viewed — regardless of whether that user had chosen to display only their country, only their city, or nothing more granular than that on their public profile. The coordinates were stored with six decimal places, which is roughly 0.11 metre accuracy — I could get any user's home address basically.

Bug Discovery

I was studying the book Android Security Internals written by my boss back then, which led me to start Android hacking. I was browsing the app's data directory on my own Android 8 device, saw an `htbackup/` folder, opened the unencrypted SQLite file inside it, and the schema made the problem obvious. The `userbase` table had four columns: `uid`, `uptime`, `data`, `location`. The `data` blob was JSON containing the in-app username and bio, so each row was trivially attributable to a real account. The `location` blob was JSON with `b` and `c` fields holding latitude and longitude as strings, to six decimal places.

The mismatch was the interesting part. On the public profile of a user who had set their location to display *only* their country (e.g. "Japan"), the local database still received the precise lat/long that the profile page never showed. The privacy choice the user made in the UI did not propagate to the data the app cached on the devices of everyone who viewed them.

I verified this with a second test account. After deleting that test HelloTalk account, my profile was no longer visible in the app, but the precise coordinates were still sitting in my

local backup file from when I had visited the profile months before. Plugging the lat/long into Google Maps placed me at a single residential building in Kanagawa, Japan, showing my exact address at that time.

The Exploit

There is no exploitation primitive to describe — it is just "open the local SQLite file and read a column". The attack model is:

1. An attacker uses HelloTalk normally and views the profiles of users they want to track.
2. The app silently caches each viewed profile, including the precise GPS coordinates, into `htbackup/`'s SQLite database on the attacker's device.
3. The attacker reads the database off their own device (no root needed on Android 8 if you have ADB backup access).
4. For every viewed user who has *any* location set on their profile — even "country only" — the attacker now has lat/long to ~0.11 metre precision.

The profile rows from the email I sent to HelloTalk illustrate the shape. The `data` JSON identifies the user by their in-app handle; the `location` JSON gives:

```
{ "a": 3301917, "b": "35.322558", "c": "139.544225", "d": "日本", "e": "", "f": "
```

`b` is latitude, `c` is longitude. The user's public profile only showed "Japan" and the city; the precise coordinates were never shown anywhere in the UI.

Two distinct problems are stacked here. The root cause is server-side over-disclosure (CWE-359): the API hands full-precision lat/long to every viewer regardless of whether the profile owner opted into country-only, city-only, or no location at all on their public profile. The surface that made it trivially extractable is client-side cleartext storage (CWE-312): the viewer's HelloTalk client caches that over-disclosed data into an unencrypted SQLite database in `htbackup/`. Encrypting the local backup, as the vendor eventually did, addresses the second layer but does not change the underlying design issue that the server still hands the precise coordinates over before they can be cached at all. The cleanest fix would have been to never ship the precise coordinates down to viewer devices

in the first place, and to have the server return only the granularity the profile owner had opted into. I did not retest whether the post-fix behaviour also stopped serving the precise coordinates to the client.

Afterthoughts

The interesting part of this one was the coordination. I reported it on 16 April 2019. The vendor confirmed the report quickly and pushed a fix to the local backup format some time around August 2019, but never published an advisory, did not want to assign a CVE on their side, and stopped responding once I asked about disclosure.

In September 2020 I came back to ask whether I could publish a writeup, since the issue was no longer reproducible. The thread went in circles: I said it appeared fixed, they said it was not fixed (I guess they only had encrypted backups and not fixed the root cause at that time), I said I had verified it was no longer reproducible. But they insisted that I do not share it, and then the thread quietly died after doing in a few more circles. I switched to Chinese halfway through to make sure nothing was being lost in translation, but the answer did not change.

I then requested a CVE from MITRE, which took nearly 3 years to assign, and got CVE-2020-25900 in the **RESERVED** state on 12 August 2023. I never got around to publishing the public reference, partly because the vendor relationship had soured and partly because life got in the way, and the CVE has been sitting in RESERVED ever since.

This blog post is, finally, that public reference. The fix shipped years ago, the vulnerable version is well out of support, and there is no live exploitation primitive left here — it is now purely a historical record so the CVE record can move out of RESERVED state.

If you are a maintainer reading this: please, when a reporter asks “is it okay if I publish this”, “we are still working on it” is not a useful answer once the fix has shipped. A short coordinated-disclosure note from your side is much better than a six-year radio silence that leaves the security record in an indeterminate state.

Timeline

2019-03-30 JST — Discovered the unencrypted `htbackup/` SQLite database during on-device review

2019-04-16 19:28 JST — Reported to vendor (hi[at]hellotalk.com) with PoC row and Google Maps lookup

2019-04-16 19:39 JST — Vendor acknowledged, asked for platform (Android)

2019-04-17 11:07 JST — Vendor confirmed report had been passed to engineering

2019-08-?? JST — Patch deployed (local backup encrypted; no public advisory issued)

2020-09-16 19:17 JST — Followed up requesting permission to disclose publicly and apply for CVE

2020-09-16 19:56 JST — Vendor responded that the issue was “not fixed completely” and asked me not to share

2020-09-18 12:32 JST — Followed up in Chinese asking for an estimated fix date, no concrete date received

2020-09-?? JST — Requested CVE assignment from MITRE via cveform.mitre.org

2023-08-12 04:21 JST — CVE-2020-25900 assigned by MITRE in RESERVED state

2026-06-05 JST — Blog post published as the public reference for the RESERVED record

Links

- CVE-2020-25900
- HelloTalk
- HelloTalk Android

Share: Twitter, Facebook

root@isopach.dev