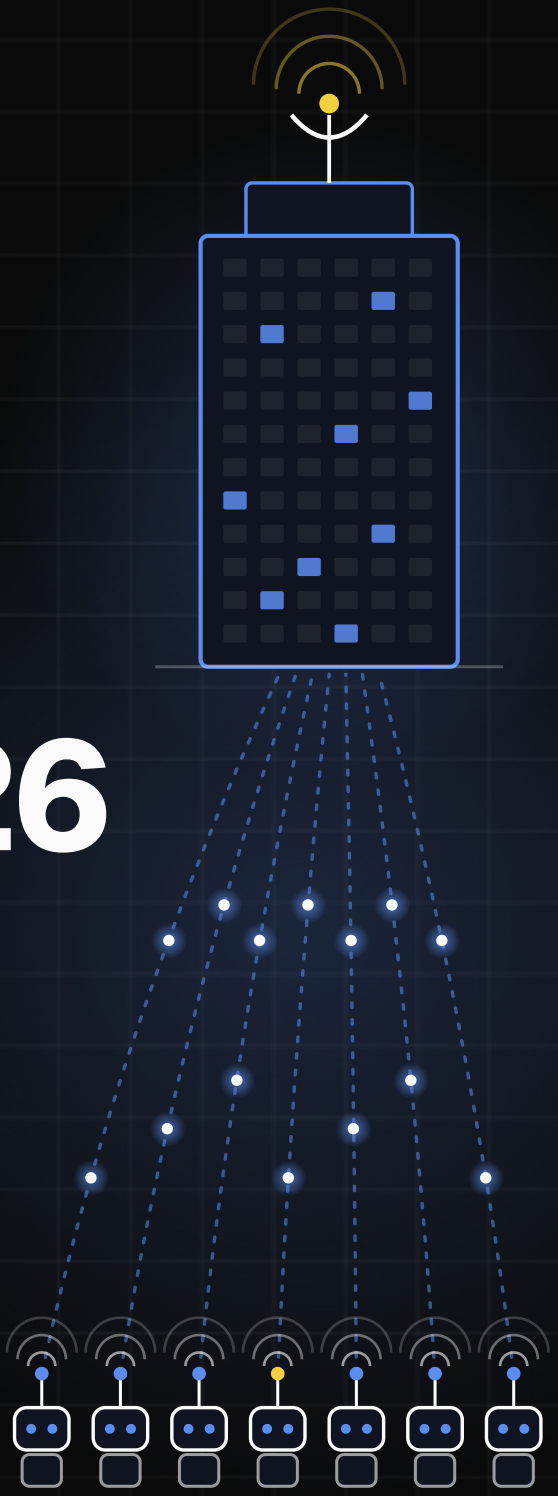
StrideNote

Stridenalysis Reports, edition 1



# Local AI Telemetry Report 2026

*Which local AI tools phone home, and what they send.*

**7**
tools examined

**11**
published investigations

**1**
test machine

September 2026

StrideNote.net is an independent research publication on modern technological systems. We follow the stories, run the tools, read the code and publish the evidence.

Edition 1

# About this report

This is the first telemetry report from Stridenalysis, the research desk of StrideNote. It brings together eleven pieces we published between June and August 2026 on one question: what does a local AI tool send from the machine it runs on?

**Publisher**

StrideNote.net

**Editor**

StrideNote Studio

**Research lead**

Hillary

**Research desk**

Stridenalysis

**Edition**

1, September 2026

**Status**

Published 19 September 2026

**DOI**

10.5281/zenodo.22848632

**Licence**

CC BY 4.0

**Dataset**

local-ai-telemetry-2026.csv, released with this report

**Contact**

info@stridenote.net

### Scope

Seven tools that sit in a working local AI stack: the OpenCode and Pi coding agents, the LM Studio model server, the huggingface_hub library that sits under most model downloads, the OpenJarvis agent framework, the Graphify knowledge-graph tool, and Visual Studio Code. All tests ran between 8 July and 8 August 2026.

### Test machine

One test machine: an Apple M4 Pro with 48 GB of unified memory, running macOS. Every measurement in this report is our own, taken on that machine.

### What this report is not

It is not a ranking and it is not a security audit. It records what we observed, how strongly we observed it, and what we did not test. A tool that is not in this report has not been cleared by us. It has simply not been tested yet.

### How to cite

StrideNote (2026) *Local AI Telemetry Report 2026*. Edited by StrideNote Studio. Stridenalysis Reports, edition 1. Zenodo. https://doi.org/10.5281/zenodo.22848632

### Licence



### Independence

No company named in this report paid for it or reviewed it before publication. The underlying articles remain online at the addresses listed in Appendix B, with their own dates.

### Corrections

Where a retest changed a first reading, the underlying articles say so, and Section 5 lists those changes. If you find an error in this report, write to info@stridenote.net and the correction will appear in the next edition.

# Contents

# Why we publish this research

Local AI rests on a simple promise. When the model runs on your own machine, your prompts stay there. That promise holds for the model. It says nothing about the software wrapped around it.

Every tool in this report makes some claim about privacy, in a settings screen, a documentation page or a privacy policy. A claim is not evidence. So we did what anyone with a laptop can do, and did it carefully. We installed each tool, read its code, listed its network connections and, where we could, captured what it sent.

We publish the full record, including what a first pass missed and how a retest changed the finding. Section 5 sets those out. A report that shows how each finding was reached can be checked, not just believed.

*Local model does not mean local software.*

StrideNote, OpenCode vs OpenJarvis, 8 July 2026

This is the first edition. It covers seven tools on one machine over one month of testing. The next edition will cover more tools, over longer windows, with the version recorded every time. Section 9 lists the tests already planned.

Everything here comes from our own tests, our own published articles and primary documents from the companies and projects involved. The dataset behind every chart is published with the report, so anyone can rebuild our figures or argue with them.

Summary

# Local model does not mean local *software.*

A model that runs on your own hardware keeps your prompts on your machine. The software around it may not. We installed seven tools from a working local AI stack, read their code, listed their network connections and, where we could, captured what they sent. These are the findings.

| **2** of 4 | **0** | **1** of 3 | **2** of 5 |
|---|---|---|---|
| **Reported home** | **Prompts seen leaving** | **Working off switch** | **Permanent ID on disk** |
| Without being asked | In any capture | And it is undocumented | Written at install |

## 1 Two tools report to a server without being asked

OpenJarvis sends a usage event to a PostHog analytics host during use. OpenCode opens a connection to its Sentry crash-reporting endpoint at launch, before anything is typed. Both are on by default.

## 2 No capture showed prompts leaving the machine

The one OpenJarvis event we captured held counters, timings and a hashed model name. It had no message content field. We did not capture an OpenCode payload, so for OpenCode this finding is open.

## 3 Off switches are missing, hidden or pointed at the wrong thing

OpenJarvis has a working switch that its own documentation never names. OpenCode has none among 41 disable flags. The Hugging Face library documents a switch for a telemetry function that nothing in the library calls.

## 4 The data trail that matters is often ordinary

LM Studio holds no telemetry settings and opened no outbound connection at rest, but by default routes model search and download through its own servers. The Hugging Face library reports your Python and PyTorch versions in a header on every request, with no switch.

## 5 Alarming names are poor evidence

The crash reporter in three apps had no upload address. Fifteen hits for "amplitude" were an SVG attribute. Graphify, the tool we trusted least, opened no connection beyond the local model server in 358 socket samples.

## 6 One install ID was sent, and four identifiers sit on disk

OpenJarvis keys each event to an anonymous install ID, which our capture confirmed. Visual Studio Code writes three installation identifiers and OpenCode writes one. OpenCode also connects at launch, but we have not shown that its identifier is sent.

# How much left the machine?

**4 of 13**

findings in our dataset show data sent to a reporting service without the user asking. None of our captures showed a prompt or a file leaving the machine.

**Our findings include**

- Connections we saw a tool open, and the one request stream we captured byte for byte.
- Reporting code, compiled-in addresses and settings we read in the shipped builds.
- Identifiers each tool wrote to disk.

**Our findings do not include**

- **What a payload held when we did not capture it.** Why? A connection proves contact with a host, not what travelled over it.
- **Anything sent outside our test windows.** Why? Socket listings are snapshots, and a tool that reports every few hours would not appear in one.
- **Crash reports.** Why? We did not induce a crash in any tool.
- **Other versions, or Windows and Linux builds.** Why? Every test ran on one Mac, on the version installed at the time.

The 13 findings are the rows of Appendix A. Figure 5 in Section 3 breaks them down by outcome.

# We tested the tools we trust, not only the ones we doubted

We started with a research agent we had reason to question and ended with the coding agent in our own stack.

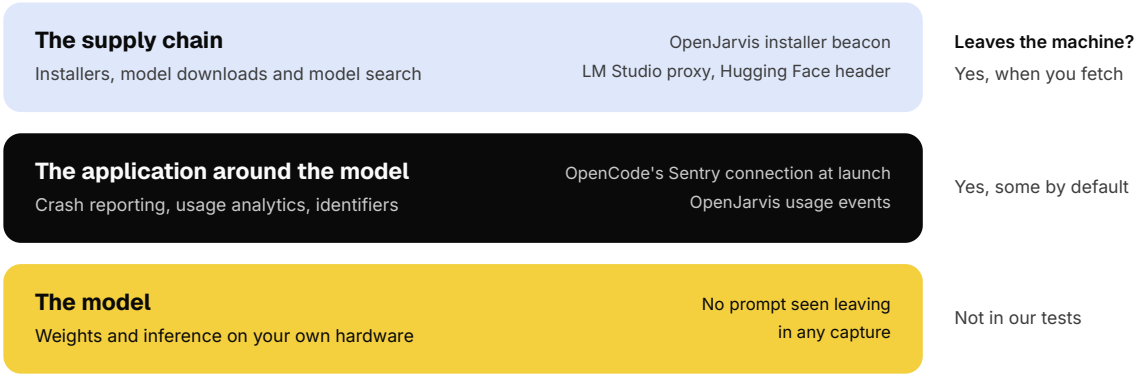
StrideNote runs its AI work on local models. A prompt that never leaves the machine cannot be read, kept or trained on by anyone else. Our method guide on local AI privacy (StrideNote, 2026a) calls this demonstrated privacy: a claim you can check with a network monitor rather than a promise you have to accept.

That guarantee covers the model. It says nothing about the application wrapped around it, or about the installers and libraries that fetch models. Figure 1 shows the three layers. The model sat where we expected. The other two layers opened connections of their own.

Figure 1

**Where data can leave a local AI tool**

The model is one layer of three. In our tests, the layers around it are where connections came from.



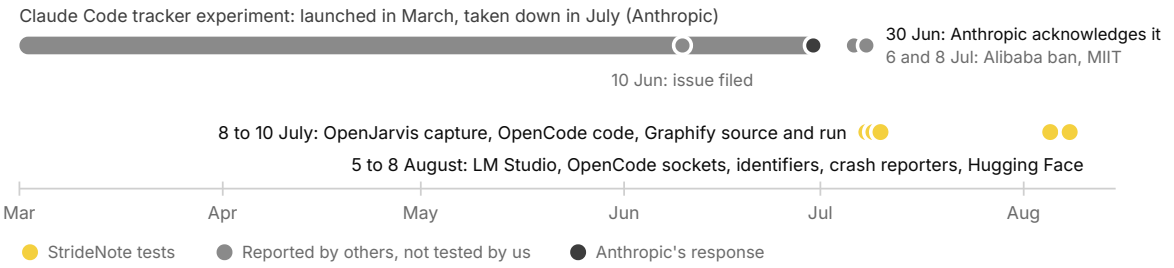

Source: StrideNote analysis of the tests in this report, July and August 2026 StrideNote, 2026

In July 2026 we tested OpenJarvis, a research agent from Stanford's Scaling Intelligence Lab, and found analytics switched on by default (StrideNote, 2026b). We then ran the same inspection on OpenCode, the coding agent in our own stack, and found a crash-reporting endpoint we could not switch off (StrideNote, 2026c). The rest of this report follows from that result. If the tool we trusted had not been checked, neither had the others.

Figure 2

### The tests in time

Our tests ran in two blocks, in July and August 2026. The Claude Code case in Section 6 broke in the same weeks.




Source: StrideNote tests; anthropics/claude-code (2026); Shihipar (2026); CNBC (2026a, 2026b). Test dates where stated, otherwise publication dates

StrideNote, 2026

## The tools in this report

| Tool | What it is | Status in our testing |
|---|---|---|
| **OpenCode** | Desktop coding agent, an Electron app | In use |
| **Pi** | Terminal coding agent | In use |
| **LM Studio** | Local model server and desktop app, an Electron app | In use |
| **huggingface_hub** | Python library behind most model downloads | Installed, version 1.17.0 |
| **OpenJarvis** | Open-source personal agent framework from Stanford | Tested on 8 July 2026, then removed |
| **Graphify** | Turns a code folder into a knowledge graph | Tested on 10 July 2026, not adopted |
| **Visual Studio Code** | Code editor, an Electron app | Installed |

The seven tools were chosen because we run them, not as a sample of the market. Section 8 sets out what that means for the results.

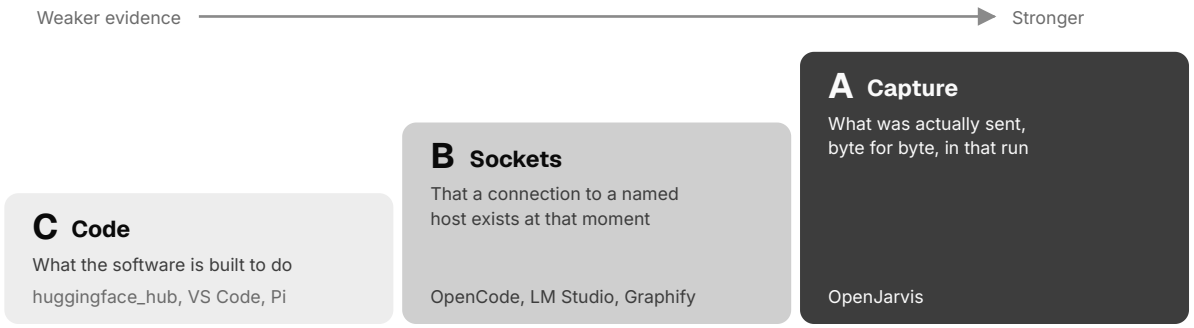
# A claim about software is worth the check behind it

We grade every finding by how directly we observed it, from reading the code to capturing the bytes.

Figure 3

### The evidence ladder

Each tool sits on the highest step we reached for it. Only OpenJarvis reached a full capture.




Source: StrideNote tests, July and August 2026

StrideNote, 2026

| Grade | What we did | What it proves | What it does not prove |
|---|---|---|---|
| A Capture | Pointed the tool's reporting host at a listener on the same machine, ran the same task with the setting on and off, and recorded every request | What was sent, byte for byte, in that run | What another version, task or configuration would send |
| B Sockets | Listed every network connection held by every process of the running tool, and resolved the addresses | That a connection to a named host exists at that moment | What travels over it; anything sent outside the sampling window |
| C Code | Read the source or searched the shipped bundle, settings files and files written to disk | What the software is built to do | That it does it; code can be present and never run |

## The rules we held ourselves to

A search count is not a finding. A string in a bundle shows that a byte sequence is present. Only the code around it shows what it does. Section 5 records the leads that failed this test.

Presence is not transmission. A crash reporter, an analytics library or an identifier on disk is only a finding once there is an address it reports to, and the strongest evidence is a connection to that address.

Grade C findings are stated as what the code says. Where we could not capture a payload, the report says so in the same sentence as the claim. Dates are the test date where the underlying article states it, otherwise its publication date.

# Four of 13 findings sent data to a reporting service without being asked

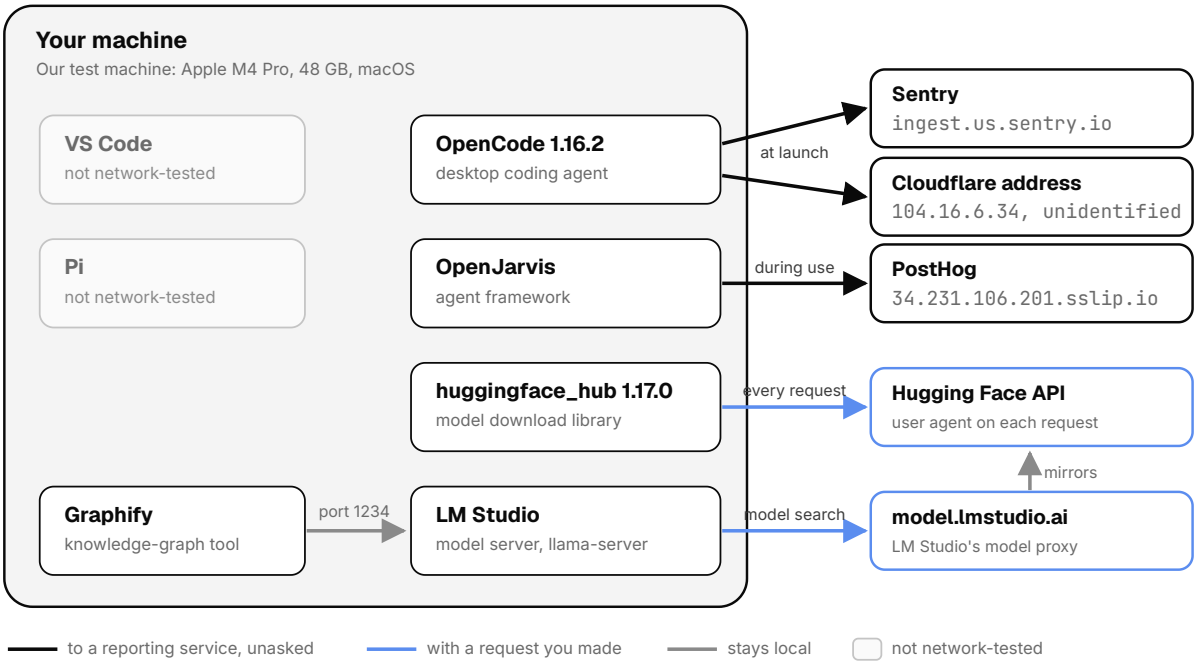
Every line that crosses the edge of the machine in Figure 4 is something we found leaving it.

**Colours in this report** ● Sent to a reporting service without being asked ● Sent as part of a request you made ● Nothing left the machine ● Stored on disk only, or not tested

Figure 4

### Where each tool connects

Black lines go to reporting services. Blue lines carry requests you made. Graphify talks to nothing but LM Studio on the same machine.

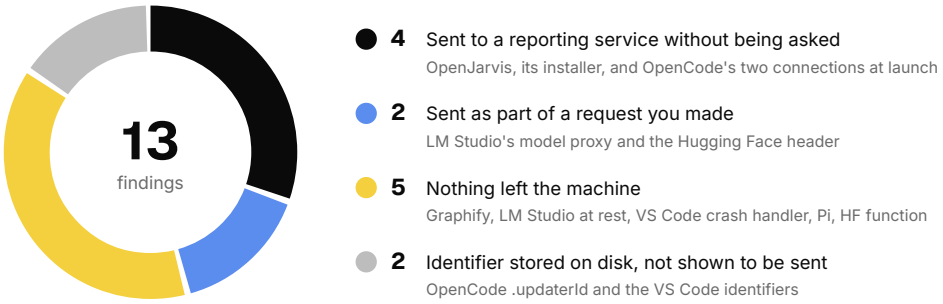



Source: StrideNote (2026b to 2026j), tests from 8 July to 8 August 2026 StrideNote, 2026

Figure 5

## The 13 findings, by outcome

Each finding is one row of Appendix A. Five showed nothing leaving the machine.




Source: Appendix A; StrideNote tests, 8 July to 8 August 2026

StrideNote, 2026

Figure 6

## Outcome against evidence

Only one finding rests on a full capture. The OpenCode findings rest on connections seen, not content, which is why Section 9 puts an OpenCode capture first.

| | Reported home, unasked | Sent with your request | Nothing left | On disk only |
|---|---|---|---|---|
| **A: Capture** bytes seen | OpenJarvis | | | |
| **B: Sockets** connection seen | OpenCode: Sentry<br>OpenCode: Cloudflare | LM Studio proxy | Graphify<br>LM Studio at rest | |
| **C: Code** code read | OpenJarvis installer | Hugging Face header | HF telemetry function<br>VS Code crash handler<br>Pi session IDs | OpenCode .updaterId<br>VS Code IDs |

Source: Appendix A

StrideNote, 2026

Figure 7

## Telemetry fingerprints of seven tools

One card per tool. Black marks a problem, yellow a clean result, blue data sent with your own request, grey what we could not test or only found on disk.

### OpenJarvis — A

Tested, removed

- Reports home unasked: Yes, one event per session
- Your content leaving: No, confirmed by capture
- Off switch: Works, but undocumented
- Install ID: Sent with each event

### OpenCode — B

In use, desktop 1.16.2

- Reports home unasked: Yes, Sentry at launch
- Your content leaving: Not captured
- Off switch: None in 41 flags
- Install ID: On disk, .updaterId

### LM Studio — B

In use

- Reports home unasked: No; model search goes via its proxy
- Your content leaving: No, inference on localhost
- Off switch: useHFProxy setting, untested
- Install ID: None found

### huggingface_hub — C

Installed, 1.17.0

- Reports home unasked: No; a header on every request
- Your content leaving: No
- Off switch: None for the header
- Install ID: None in the cache

### Graphify — B

Tested, not adopted

- Reports home unasked: No, 0 remote connections
- Your content leaving: Only to the local model
- Off switch: Nothing to switch off
- Install ID: Not checked

### VS Code — C

Installed

- Reports home unasked: Not network-tested
- Your content leaving: Not tested
- Off switch: Not tested
- Install ID: 3 on disk, storage.json

### Pi — C

In use

- Reports home unasked: Not network-tested
- Your content leaving: Not tested
- Off switch: Not tested
- Install ID: None; session IDs only

### How to read a card

The letter is the evidence grade from Section 2

- Problem found
- Sent with your own request
- Clean result
- Not tested, or on disk only

Source: Appendix A; StrideNote (2026b to 2026j)

StrideNote, 2026

# Two tools report home by default, and only one can be switched off

## OpenJarvis sends one usage event per session, with an off switch nobody documented

**Tested**

8 July 2026, desktop build tagged desktop-v1.0.2 and the command line

**Evidence**

A A/B capture at a local listener

**Endpoint**

PostHog host at 34.231.106.201.sslip.io, hard-coded project key

**On by default**

Yes

**Off switch**

Works, documented nowhere

**Sources**

StrideNote (2026b, 2026c)

OpenJarvis ships external analytics enabled. The posthog library is a runtime dependency of both of its Python environments, and its configuration carries a hard-coded project key and host, flushed every 30 seconds.

The project's `docs/telemetry.md` is better than most. It describes two redaction layers and states that no chat content, prompts or file paths are sent. It opens by promising to explain how to opt out and contains no opt-out section. The one control it names, `jarvis analytics reset-id`, does not exist in the command line we tested.

Case study

### The pineapple test

We redirected the analytics host to a listener on 127.0.0.1, started the server and asked one question whose expected answer was the word "pineapple". Exactly one request arrived. We then set `enabled = false` under an `[analytics]` heading in `~/.openjarvis/config.toml`, restarted and sent the identical question to the identical model. No request arrived. Figure 8 shows the one request.

Figure 8

## The one event OpenJarvis sent

Field names as captured. We list what each field describes rather than its values. The prompt never appeared.

`POST /batch/` to the PostHog host, redirected to our listener — Captured 8 July 2026, analytics on

| | |
|---|---|
| `event` | chat_session_ended |
| `distinct_id` | anonymous install UUID |
| `turn_count` | turns in the session |
| `tokens_in, tokens_out` | token counts |
| `latency p50, latency p95` | response times |
| `tool_count, unique_tools, unique_models` | tool and model use |
| `error_count, duration_ms` | errors and session length |
| `model_hash` | hashed model name |
| `OS, Python version` | environment |
| `library` | posthog-python 7.22.0 |
| `$geoip_disable` | true |

Not in the payload: a message content field of any kind. The prompt word "pineapple" appeared 0 times.

**1**
request with analytics at the shipped default

**0**
requests with `enabled = false`, same question, same model

Source: StrideNote local beacon capture, 8 July 2026 — StrideNote, 2026

Figure 9

## OpenJarvis from install to use

What left the machine at each stage, in the order it happened on our machine.

### Install
Before any question is asked

- Install progress beacon. The gate is analytics_enabled() { return 0; }, so it always reports
- Background model downloads: qwen3.5 at 0.8b, 2b, 4b and 9b, unless you pass --no-bg-orchestrator

### First launch
Desktop app starting its server

- The app waited on port 8000, held by another program, with no error shown. No data finding

### Every session
During normal use

- One chat_session_ended event, flushed on a 30 second cycle
- With enabled = false: 0 requests

### Hours later
After we thought it had stopped

- Ollama resumed a pull nobody asked for: 5 connections to the model registry, one partial blob reaching 16 GB at about 32 MB/s

● Sent to a reporting service unasked ○ Other network activity you did not ask for ● Nothing left ● No data finding

Source: StrideNote (2026b), 8 July 2026 — StrideNote, 2026

**Check yours.** Open `~/.openjarvis/config.toml` and look for an `[analytics]` heading. If `enabled = false` is missing, add it and restart. Install with `--no-bg-orchestrator` to skip the background downloads.

## OpenCode contacts Sentry at launch, and none of its 41 flags turns it off

**Tested**

Desktop 1.16.2. Bundle read 8 July 2026; sockets in the audit published 8 August 2026

**Evidence**

**B** Sockets at launch, plus bundle reading

**Endpoint**

Sentry, ingest.us.sentry.io, DSN compiled into the app

**On by default**

Yes, unconditional at startup

**Off switch**

None found

**Sources**

StrideNote (2026c, 2026d)

OpenCode's code sits in a 93 MB `app.asar` archive. At renderer startup, in a block with no environment check and no configuration lookup, it initialises Sentry with a production DSN, the environment set to production and the release set to `desktop@1.16.2`.

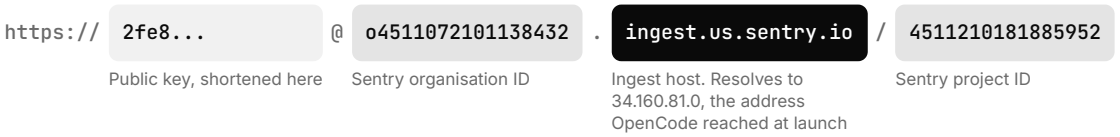
```
init$1({
  dsn:
"https://2fe8...@o4511072101138432.ingest.us.sentry.io/4511210181885952",
  environment: "production",
  release: "desktop@1.16.2",
  integrations: (integrations) => integrations.filter(i => i.name !==
"Breadcrumbs")
})
```

We launched the app and listed every socket held by its five processes. Two outbound connections were established with no prompt typed and no project open. The first, 34.160.81.0 on port 443, is the address the DSN's host resolves to. The second, 104.16.6.34, sits in Cloudflare's address space and does not match opencode.ai. We did not identify it.

Figure 10

### Anatomy of the address compiled into OpenCode

A DSN names one Sentry project, so its presence in a shipped build means the developer wired it in.

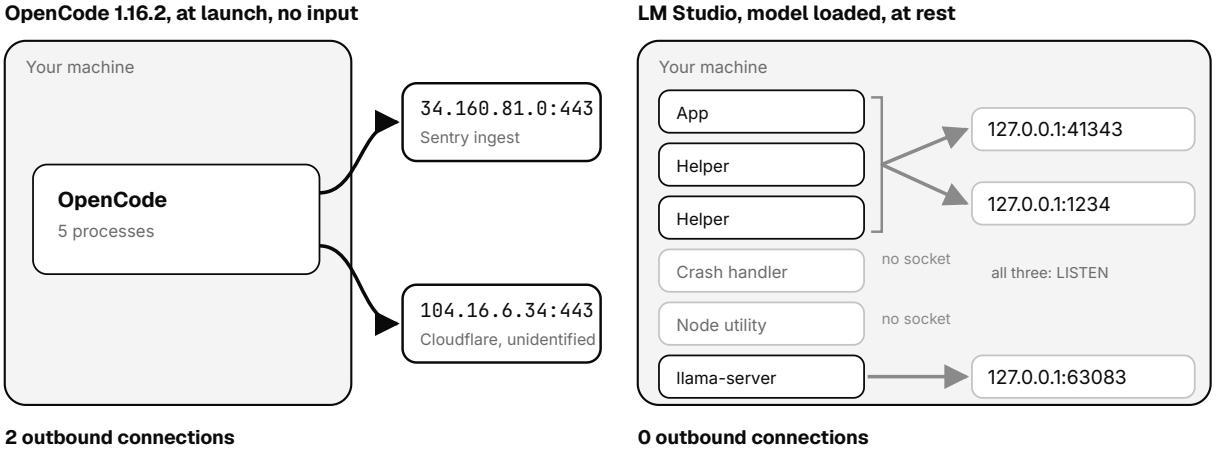



Source: OpenCode desktop 1.16.2 app.asar, read 8 July 2026 StrideNote, 2026

Figure 11

### Two launch graphs

What each tool's processes held, from socket listings. OpenCode reached out before any input. LM Studio only listened on its own machine.




Source: StrideNote (2026d, 2026e). The two sockets listed as "LM Studio" are shown against the app and helpers together StrideNote, 2026

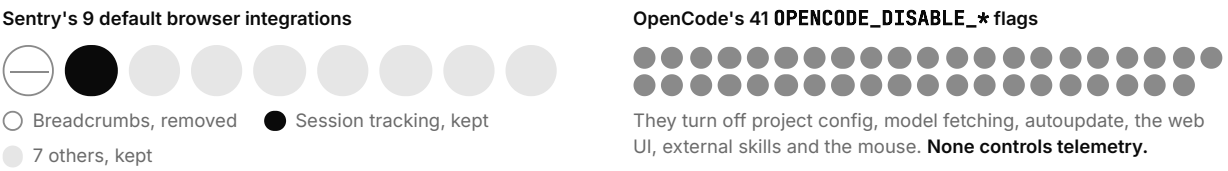
### What it would carry

OpenCode's filter removes one integration, Breadcrumbs, which is the one that records user actions before an error. That is a deliberate privacy choice. The integration that tracks sessions stays on, which by Sentry's documented defaults means a session report at launch (Sentry, 2026a). The `sendDefaultPii` option is never set, and Sentry gives its default as false (Sentry, 2026b). An error event still carries a stack trace, and in a coding agent a stack trace is the payload most likely to hold a file path.

Figure 12

**What OpenCode removed, and what it lets you switch off**

One integration of nine removed. Zero of 41 flags for telemetry.




Sources: Sentry (2026a); StrideNote (2026c)

StrideNote, 2026

Limit: we did not capture a payload. Doing so means repointing a DSN compiled into a signed archive, or inducing an error with a local sink in place. That is the first test in Section 9.

**Check yours.** Launch OpenCode, type nothing, and list the connections its processes hold. An established connection to an address that resolves from `ingest.us.sentry.io` is the crash reporter at work.

## LM Studio keeps inference local and routes model search through its own proxy

**Tested**

Running install with a model loaded; audit published 5 August 2026. Version not recorded

**Evidence**

**B** Sockets across six processes, plus settings

**Endpoint**

model.lmstudio.ai, for model search and download

**On by default**

The proxy, yes

**Off switch**

The useHFProxy setting; the effect of turning it off was not tested

**Source**

StrideNote (2026e)

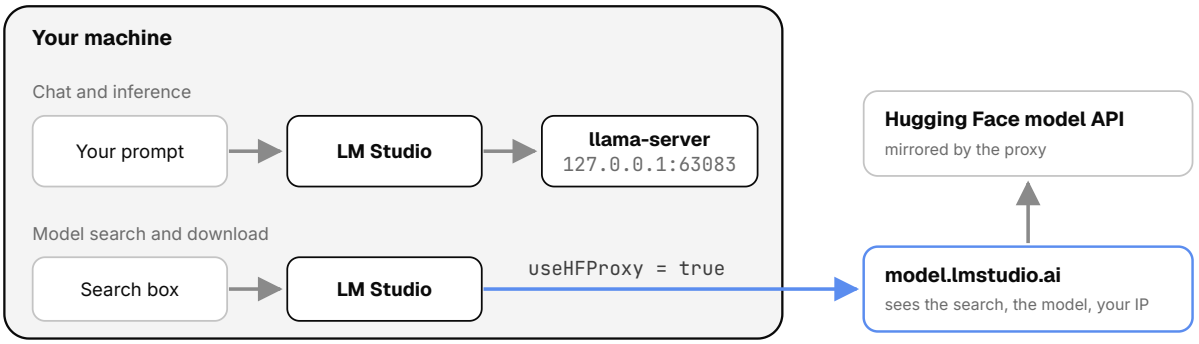
We listed the network connections of all six LM Studio processes: the app, two helpers, the crash handler, an internal Node utility and the llama.cpp server it spawns. There were three sockets, all listening on localhost, and no established outbound connection. Figure 11 shows them.

The settings file holds 28 keys. None is named for analytics, telemetry, usage reporting or tracking. The privacy policy says conversations, histories and documents stay local (LM Studio, 2026a), and nothing we found contradicts it.

Figure 13

### Two paths through LM Studio

Inference stays on localhost. On a default install, model search and download go to LM Studio's own proxy.




Source: StrideNote (2026e) StrideNote, 2026

The proxy answered the same requests as the Hugging Face model API when we tested it. Its operator therefore sees which models you look for and pull, and the address doing the looking. The privacy policy discloses that IP address and basic device information reach a CDN during search, download and update checks. This is routing, not telemetry, and it is disclosed. It still creates a record of what you are building.

Limit: a point-in-time connection listing. A tool that reports every few hours would not appear in it. The crash handler held no connection, and we did not induce a crash.

**Check yours.** Open `~/.lmstudio/settings.json` and look for `useHFProxy`, `hfSearchToken` and `hfDownloadToken`. A populated token identifies your account, not only your IP address.

## The Hugging Face library reports your Python and PyTorch versions on every request

**Tested**

huggingface_hub 1.17.0; published 8 August 2026

**Evidence**

C Source reading and header construction

**Sent**

A user agent on every request

**On by default**

Yes

**Off switch**

None for the header; offline mode stops all requests

**Source**

StrideNote (2026f)

The library contains a 125-line telemetry module that defines `send_telemetry()`. It posts to a telemetry path on the hub, runs on a background thread, and returns at once if offline mode or `HF_HUB_DISABLE_TELEMETRY` is set. Both controls are documented (Hugging Face, 2026a). Nothing in the library calls the function. It exists for downstream libraries to report their own usage.

What does leave is a header. Building the default request headers on our machine produced the user agent in Figure 14.

Figure 14

### What the Hugging Face library says about your machine

The user agent built on our machine, field by field, and the two reporting paths compared. No token file was present, so our requests were unauthenticated.

| `unknown/None` | `hf_hub/1.17.0` | `python/3.13.2` | `torch/2.10.0` | `agent/unknown` |
|---|---|---|---|---|
| Calling app, not set here | Library version | Python, down to the patch release | PyTorch version, looked up on your machine when the header is built | Agent field, not set here |

**send_telemetry()**

Documented, opt-out with `HF_HUB_DISABLE_TELEMETRY`. Called by nothing in the library.

**The request header**

Undocumented as telemetry, sent on every API call. No switch. Only `HF_HUB_OFFLINE` stops it, by stopping all requests.

Source: StrideNote (2026f)

StrideNote, 2026

**Check yours.** Look for a token file in the hub cache; a stored token attributes every download to an account. Set `HF_HUB_DISABLE_TELEMETRY=1` if you use libraries built on the hub, such as transformers or diffusers. Only `HF_HUB_OFFLINE` stops the header.

## Graphify, the tool we trusted least, talked only to the local model

**Tested**

Source read 9 July 2026; run on 10 July 2026 against LM Studio with Gemma 4 31B. Version not recorded

**Evidence**

**B** 358 socket samples during a model pass

**Endpoint**

None beyond the configured model server

**Off switch**

Nothing to switch off

**Sources**

StrideNote (2026g, 2026h)

Graphify's package is named `graphifyy` and installs a binary called `graphify`, which is the shape a typosquat takes, and our own supply-chain check refused the first install. We read nine files, 493 KB of its Python source, before running it. We found no analytics vendor names and no telemetry code. The hard-coded addresses are the APIs of language-model providers you would have to select, plus export.arxiv.org and a social-post embed service used when you ask it to fetch those sources.

The one thing that pushes toward the cloud is a tip printed at the end of each rebuild, which suggests a Gemini API key. The local options work, and the tool does not mention them at that point.

Figure 15

### Graphify on a local model

Each dot is one sample of every socket Graphify held. None showed an outside address.

**358 socket samples during the labelling pass**, each one showing LM Studio on localhost:1234



| | Code pass | Labelling pass |
|---|---|---|
| **Model used** | None, tree-sitter | Gemma 4 31B in LM Studio |
| **Time** | 1.5 s | 112.4 s |
| **Tokens** | 0 | 2,069 in, 1,312 out |
| **Remote connections** | **0** | **0** |

Source: StrideNote hands-on test, 10 July 2026 StrideNote, 2026

**Check yours.** Install it by name, `graphifyy`, into an isolated environment. Point `OPENAI_BASE_URL` at your local model server, and watch its sockets during the first run.

## Two of five tools write a permanent identifier to disk

**Tested**

Five tools; published 8 August 2026

**Evidence**

C Files written to disk

**Result**

2 of 5 write a permanent installation identifier

**Source**

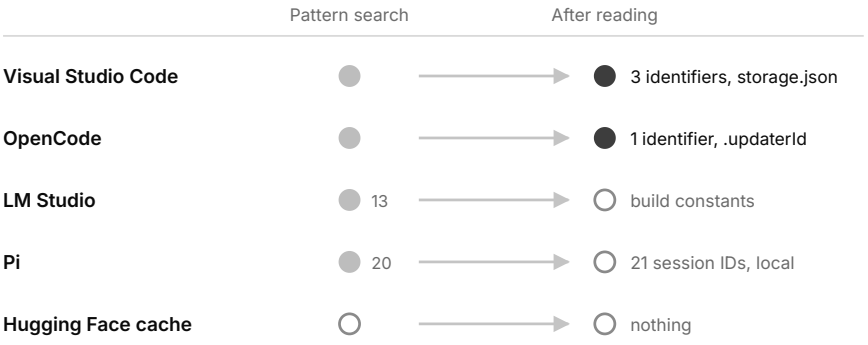
StrideNote (2026i)

An IP address changes and is shared. An identifier written once to disk and read at every launch turns separate requests into one profile of one machine. We searched what five tools write to disk for identifier-shaped values. A pattern search flagged four of the five. Reading the values reduced that to two.

Figure 16

### Flagged, then read

A filled dot on the left is a pattern match, with the count where the article gives it. A dark dot on the right is a real installation identifier.




Source: StrideNote (2026i) StrideNote, 2026

Visual Studio Code's three identifiers are `telemetry.machineId`, `telemetry.sqmId` and `telemetry.devDeviceId`, documented by Microsoft (Microsoft, 2026). OpenCode's is a 36-character value created at first run and unchanged since. OpenJarvis was not in this sweep; its install ID appears in the event we captured (Figure 8).

## Three crash reporters, and none has an upload address

**Tested**

LM Studio, OpenCode, Visual Studio Code; published 8 August 2026

**Evidence**

C Bundles, plus B sockets for LM Studio

**Result**

No upload address configured in any of the three

**Source**

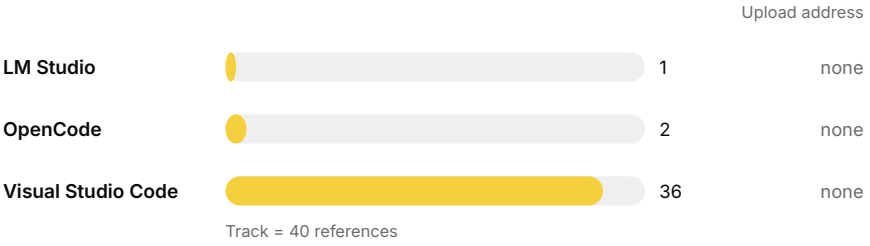
StrideNote (2026j)

All three are Electron apps, and each ships Chromium's crash handler, `chrome_crashpad_handler`. Electron's crash reporter sends nothing unless the app starts it with a `submitURL` (Electron, 2026a). In all three bundles the parameter names appear, because the framework code is bundled, but no URL is assigned to them. LM Studio's handler held no sockets throughout our inspection.

Figure 17

### References to the crash reporter in each bundle

More code did not mean more reporting. None of the three had an upload address.




Source: StrideNote (2026j) StrideNote, 2026

OpenCode's error reporting runs through the Sentry JavaScript library instead, which has no distinctively named process.

Limit: we did not induce a crash. The handler writes crash dumps locally, and an app could collect them by another route later.

# Six leads looked like telemetry and were not

Each of these looked stronger than the finding that survived. We record them because the method depends on catching them.

| Lead | What it looked like | What it was |
|---|---|---|
| **"amplitude" in OpenCode** | 15 hits for the name of a product-analytics vendor | An SVG filter attribute in a bundled rendering library's attribute table |
| **OpenTelemetry in OpenCode** | More than 150 references, including collector definitions | A library with no exporter configured and no collector address |
| **IDs in LM Studio** | 13 identifier-shaped values | Constants inside a compiled binary |
| **IDs in Pi** | 20 identifier-shaped values | One session identifier per conversation, kept locally |
| **Crash handlers** | A Chrome-named process in three non-browser apps | Electron's bundled handler with no upload address |
| **Graphify** | A package name shaped like a typosquat | No telemetry in the source, and no remote connection in use |

Case study

### What the first pass missed

Our first pass on OpenJarvis (StrideNote, 2026b) needed three retests before the finding was complete. The first note listed the tool as terminal-only; a desktop app had shipped six days earlier. The desktop app then looked broken; it was waiting for a network port that another program held. The first explanation for that pointed at two configuration files; the app in fact passes the port on the command line, which overrides the file.

Each retest moved the finding closer to what the software actually does. That is why every finding in this report carries its evidence grade, and why the dataset records how each one was checked.

# Claude Code hid server and timezone checks in its prompts, then took them out

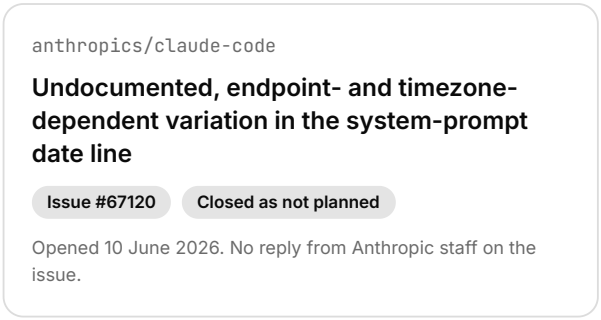
This case is reported by others. We include it because it shows the mechanism at its most deliberate, and because it was found the same way: someone read the code.

On 10 June 2026 an issue filed in the public tracker for Claude Code, Anthropic's terminal coding tool, described behaviour that appeared in no release notes (anthropics/claude-code, 2026). According to that report, lists stored behind XOR and base64 encoding held known hostnames and lab keywords. If a user pointed the tool at a non-default server, the code compared that server against the lists. It also checked whether the system timezone was Asia/Shanghai or Asia/Urumqi. A match was not logged. It was encoded in the date line of the system prompt sent with each request, using a changed date separator and visually similar Unicode apostrophes. No Anthropic staff member replied on the issue.

Figure 18

**The report and the acknowledgement**

The public issue that described the code, and the first-hand statement from an engineer on the Claude Code team.





Post on X, 30 June 2026. See the full post: x.com/trq212/status/2072079729331777817

Sources: anthropics/claude-code (2026); Shihipar (2026)

StrideNote, 2026

### What Anthropic said

On 30 June 2026 Thariq Shihipar, an engineer on the Claude Code team, acknowledged the code publicly on X. He described it as "an experiment we launched in March" (Shihipar, 2026). The stated reasons were to stop account abuse by unauthorised resellers and to protect against distillation, the practice of training a rival model on another model's outputs. He said the team had since put stronger measures in place, that the change removing the code had been merged, and that it would be fully rolled back in the next day's release. Anthropic gave CNBC the same explanation (CNBC, 2026b). Anthropic has also published its own account of distillation campaigns against its models (Anthropic, 2026b).

CNBC reported that Alibaba banned Claude Code for its employees in early July (CNBC, 2026a). Days later China's Ministry of Industry and Information Technology told users to uninstall or upgrade versions 2.1.91 to 2.1.196 (CNBC, 2026b). Our summary of the case is in StrideNote (2026k).

### The telemetry Anthropic does document

The tracker was separate from Claude Code's ordinary telemetry, which Anthropic documents (Anthropic, 2026a). Claude Code sends usage metrics, which the documentation says never include code, prompts or file paths, and error reports with secrets and file paths redacted. Each has its own switch: `DISABLE_TELEMETRY` and `DISABLE_ERROR_REPORTING`. `CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC` turns off all non-essential traffic at once. That is the pattern this report asks for: disclosed, explained, and with a working switch. The tracker had none of the three until it was found.

> Claude Code is a cloud-connected tool, not a local one, and we did not test it. Nothing in this section is a StrideNote measurement. Every data flow we measured was either documented or visible in readable code. This one was built not to be seen.

# Five checks anyone can run in minutes

Each check needs nothing beyond the tools that ship with the operating system.

### Watch the sockets at launch

Open the tool and do nothing. List the network connections its processes hold. On macOS, `lsof -nP -iTCP -a -p <pid>` does this for one process. Anything bound to 127.0.0.1 is talking to your own machine. An established connection to an outside address is a question worth answering. Resolve the address and see whose it is.

### Read the settings file

Look for keys named for analytics, telemetry or usage, and for proxy or mirror settings. In LM Studio the key to look for is `useHFProxy` in `~/.lmstudio/settings.json`. Check whether any access token is stored, because a token turns an anonymous request into an attributed one.

### Look for the destination, not the machinery

A crash reporter or an analytics library in a bundle proves nothing on its own. Search for the address it would send to: a DSN, a collector URL, a hard-coded host. No destination means nothing is sent by that route.

### Filter identifier-shaped values

If a value sits inside a compiled binary, it is a build constant. If there are many and they grow with use, they are session identifiers. One value, in a small file or settings key, created at install and unchanged since, is a machine identifier.

### Run the A/B capture when it matters

If a tool lets you set its reporting host, point it at a listener on your own machine. Run one fixed task with the setting on, then off, and compare what arrives. This is the only check in this list that shows the bytes themselves.

# What these findings do not show

The findings are narrower than the questions, and the gaps are listed here so they are not read as clean results.

- One machine, one operating system. Every test ran on one Apple M4 Pro under macOS. Behaviour on Windows or Linux builds may differ.
- Point in time. Each result describes one version on one date. Several articles did not record the exact version of LM Studio, Graphify, Visual Studio Code or Pi, and the dataset marks those fields as not recorded.
- Short windows. Socket listings and short monitoring runs miss anything that reports every few hours. Only the OpenJarvis test captured the full content of what was sent.
- No induced crashes. None of the crash-reporting paths was tested under a real crash.
- No OpenCode payload. The OpenCode findings rest on the bundle and on connections at launch, not on captured content.
- Selection. The seven tools are the ones we run. They are not a sample of the market, and the counts in the summary should not be read as rates for local AI software in general.
- Two tools were not network-tested. Visual Studio Code and Pi appear only in the disk and bundle checks.

# Seven tests that close the gaps

Each item closes a gap named in Section 8.

| Test | Why | Target grade |
|---|---|---|
| **OpenCode error capture** | Induce an error with a local sink in place and record what a real event contains | A |
| **LM Studio over a working day** | Replace a snapshot with a full day of recorded requests | A |
| **Hermes Agent** | The always-on assistant in our own stack, not yet inspected | B or better |
| **Pi on the network** | Move Pi from disk checks to socket evidence | B or better |
| **Visual Studio Code on the network** | Test whether the three identifiers leave the machine | B or better |
| **Ollama** | The runtime the OpenJarvis installer added, and a common default | B or better |
| **Version retests** | Repeat every 2026 test on the current version and record the version each time | As 2026 |

# The dataset

The same rows are published as `local-ai-telemetry-2026.csv`. Each row is one finding, not one tool.

| Tool | Version | Date | Gr. | Destination | Trigger | What is sent | Default | Off switch |
|---|---|---|---|---|---|---|---|---|
| **OpenJarvis** | desktop-v1.0.2 | 2026-07-08 | A | PostHog, 34.231.106.201.sslip.io | During use, 30 s flush | chat_session_ended: counters, timings, hashed model name, OS, Python version, install UUID. No content | On | [analytics] enabled = false; works; undocumented |
| **OpenJarvis installer** | install.sh | 2026-07-08 | C | Analytics endpoint | During install | Install progress | On | None; host can be redirected |
| **OpenCode** | 1.16.2 | 2026-08-08 | B | Sentry, ingest.us.sentry.io (34.160.81.0) | At launch; on error | Not captured. Session report and error events by SDK design; Breadcrumbs removed | On | None found in 41 flags |
| **OpenCode** | 1.16.2 | 2026-08-08 | B | 104.16.6.34 (Cloudflare) | At launch | Not identified | On | Not assessed |
| **OpenCode** | 1.16.2 | 2026-08-08 | C | File on disk | First run | .updaterId, one permanent identifier | On | Not assessed |
| **LM Studio** | Not recorded | 2026-08-05 | B | model.lmstudio.ai | Model search and download | Search queries, model paths, IP address | On | useHFProxy; effect not tested |
| **LM Studio** | Not recorded | 2026-08-05 | B | None | At rest, model loaded | Nothing; 3 localhost listeners, 0 outbound | n/a | n/a |
| **huggingface_hub** | 1.17.0 | 2026-08-08 | C | Hugging Face API | Every request | User agent with hub, Python and PyTorch versions; IP address | On | None; HF_HUB_OFFLINE stops all requests |
| **huggingface_hub** | 1.17.0 | 2026-08-08 | C | Hub telemetry path | Never; no caller | Nothing from the library itself | n/a | HF_HUB_DISABLE_TELEMETRY |
| **Graphify** | Not recorded | 2026-07-10 | B | localhost:1234 only | Labelling pass | Requests to the local model server | n/a | Nothing to switch off |
| **Visual Studio Code** | Not recorded | 2026-08-08 | C | File on disk | Install | Three identifiers in storage.json | On | Not tested |
| **Visual Studio Code** | Not recorded | 2026-08-08 | C | None configured | On crash | Crash handler present, no upload address | n/a | n/a |
| **Pi** | Not recorded | 2026-08-08 | C | None | Per conversation | Session identifiers kept locally | n/a | n/a |

Appendix B

# Sources

## StrideNote investigations


StrideNote (2026a) Local AI privacy: how to prove your data never leaves. https://stridenote.net/local-ai-privacy/

StrideNote (2026b) OpenJarvis on a Mac: we retested Stanford's local AI agent. 8 July. https://stridenote.net/openjarvis-retested-mac/

StrideNote (2026c) OpenCode vs OpenJarvis: which local AI agent phones home? 8 July. https://stridenote.net/opencode-vs-openjarvis-telemetry/

StrideNote (2026d) OpenCode contacts Sentry before you type anything, and two other findings that were not real. 8 August. https://stridenote.net/opencode-telemetry/

StrideNote (2026e) LM Studio does not track you, and it routes every model search through its own servers. 5 August. https://stridenote.net/lm-studio-model-proxy/

StrideNote (2026f) Hugging Face ships a telemetry function nothing calls, and a header that reports your PyTorch version. 8 August. https://stridenote.net/huggingface-cli-telemetry/

StrideNote (2026g) Graphify has 81,000 stars. We read the code instead of running it. 11 July. https://stridenote.net/graphify-source-inspection/

StrideNote (2026h) Graphify on a local model: we ran it and watched the network. 10 July. https://stridenote.net/graphify-local-model-test/

StrideNote (2026i) Which local AI tools give your machine a permanent name, and which only look like they do. 8 August. https://stridenote.net/local-ai-machine-ids/

StrideNote (2026j) Every local AI app ships a crash reporter, and none of the three we checked turns it on. 8 August. https://stridenote.net/local-ai-crash-reporters/

StrideNote (2026k) How AI companies hide tracking code, from Claude Code to your local tools. 11 July. https://stridenote.net/how-ai-companies-hide-tracking-code/


## Primary documents


Anthropic (2026a) Data usage, Claude Code documentation. https://code.claude.com/docs/en/data-usage

Anthropic (2026b) Detecting and preventing distillation attacks. 23 February. https://www.anthropic.com/news/detecting-and-preventing-distillation-attacks

anthropics/claude-code (2026) Issue #67120: Undocumented, endpoint- and timezone-dependent variation in the system-prompt date line. GitHub, opened 10 June. https://github.com/anthropics/claude-code/issues/67120

CNBC (2026a) Alibaba bans Anthropic's Claude Code for employees. 6 July. https://www.cnbc.com/2026/07/06/alibaba-anthropic-ai-ban-claude-china.html

CNBC (2026b) China flags Claude Code as a security threat. 8 July. https://www.cnbc.com/2026/07/08/china-anthropic-ai-claude-code-backdoor-security-threat.html

Electron (2026a) crashReporter API. https://www.electronjs.org/docs/latest/api/crash-reporter

Hugging Face (2026a) Environment variables, huggingface_hub reference. https://huggingface.co/docs/huggingface_hub/package_reference/environment_variables

LM Studio (2026a) App privacy policy. https://lmstudio.ai/app-privacy

Microsoft (2026) Telemetry, Visual Studio Code documentation. https://code.visualstudio.com/docs/configure/telemetry

Shihipar, T. (2026) Post on X for the Claude Code team, 30 June. https://x.com/trq212/status/2072079729331777817

Sentry (2026a) Integrations, JavaScript SDK. https://docs.sentry.io/platforms/javascript/configuration/integrations/

Sentry (2026b) Options, JavaScript SDK. https://docs.sentry.io/platforms/javascript/configuration/options/

# Glossary

| | |
|---|---|
| **A/B capture** | Running the same task twice, with a setting on and then off, while recording every request at a listener you control. |
| **Crashpad** | Chromium's crash-reporting system. It ships with every Electron app and sends nothing without an upload address. |
| **DSN** | The address a Sentry client reports to. It is specific to one project, so a DSN in a shipped app means the developer wired it in. |
| **Electron** | A framework for desktop apps that embeds a copy of Chromium. OpenCode, LM Studio and Visual Studio Code are built on it. |
| **Local sink** | A listener on your own machine that receives requests a tool thinks it is sending elsewhere. |
| **Machine identifier** | One value written at install and read at every launch, which lets a server link separate requests to one installation. |
| **PostHog** | A product-analytics service. OpenJarvis reports usage events to a PostHog host. |
| **Sentry** | An error and session-reporting service. OpenCode reports to a Sentry project. |
| **Session identifier** | A value created for each conversation or session, usually kept locally. Many of them, growing with use. |
| **Telemetry** | Data a program sends about its own use or health. In this report it means anything sent to a reporting service rather than to a service you asked for. |
| **User agent** | A header that describes the client making a request. The Hugging Face library adds environment details to it. |

StrideNote

# Investigating technological systems.

StrideNote.net is an independent research publication on modern technological systems. We follow the stories, run the tools, read the code and publish the evidence.

Stridenalysis is our research desk. Its investigations, reports and insights are free to read, and every report is published with the data behind it.

**Read**

stridenote.net
stridenote.substack.com

**Follow**

x.com/stridenote
linkedin.com/company/stridenote
bsky.app/profile/stridenote.net
stridenote.substack.com
youtube.com/@stridenote
huggingface.co/stridenote

**Contact**

info@stridenote.net

Local AI Telemetry Report 2026, edition 1. DOI 10.5281/zenodo.22848632. Published by StrideNote.net and edited by StrideNote Studio. This report describes software behaviour observed on one machine during specific test windows. It is provided for information only and is not security, legal or investment advice.  Product names belong to their owners, and no company named here reviewed or paid for this report. Set in Geist, Inter, Newsreader and JetBrains Mono.