UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF COLUMBIA

JOSEPH KIRCHNER,

Plaintiff,

v.

MIKE JOHNSON, in his individual and official capacity as Speaker of the United States House of Representatives;
DONALD J. TRUMP, in his individual capacity as former and current President of the United States;
PAM BONDI, in her individual and official capacity as Attorney General of the United States;
BRENDAN CARR, in his individual and official capacity as Chairman of the Federal Communications Commission;
THE UNITED STATES HOUSE OF REPRESENTATIVES;
ANTHROPIC PBC;
OPENAI, INC.;
APPLE INC.;
COMCAST CABLE COMMUNICATIONS, LLC;
METR (MODEL EVALUATION & THREAT RESEARCH);
and DOES 1-20, unknown co-conspirators,

Defendants.

Case No. 1:25-cv-02735-ACR

EXHIBIT 1: INDEPENDENT VERIFICATION PROTOCOL

Step-by-Step Instructions for the Court, Counsel, or Any Third Party to Independently Verify the Anthropic Source-Code Predicate

I. PURPOSE AND SCOPE

1. This protocol enables any reader — the Court, opposing counsel, a clerk, a journalist, or any private citizen with a computer and internet access — to independently verify, in approximately five minutes, the central forensic predicate of Plaintiff's Renewed Emergency Motion for Expedited Discovery and Plaintiff's Emergency Motion for Temporary Restraining Order: **that Defendant Anthropic PBC's own source code hardcodes the Datadog third-party telemetry endpoint, the Datadog client token, the 15-second transmission interval, the per-user targeting attribute structure (including `email`, `accountUUID`, `deviceID`,**

`subscriptionType`, and `rateLimitTier`), and the closed-loop attack-signature feedback pipeline that forwards client-observed network failures to that same Datadog endpoint.

2. The verification procedure does not require trust in Plaintiff's interpretation of the evidence. It does not require expert testimony. It does not require access to non-public information. The reader follows the commands; the reader sees the output; the reader draws conclusions from what the reader has seen with the reader's own eyes.

3. **What this protocol establishes:** That Anthropic PBC's source code, in the form publicly mirrored at the GitHub repository identified in Section IV *infra*, contains the specific hardcoded values, function definitions, and architectural relationships alleged in the Renewed Motion and the TRO Motion.

4. **What this protocol does not, by itself, establish:** That the same Datadog IP address was spoofed as the source of TCP RST injection packets targeting Plaintiff (that requires Plaintiff's pcapng captures, Exhibits D-1, D-22, D-26, D-27, D-31); that per-user GrowthBook force-rules attach Plaintiff's account specifically to the Datadog pipeline (that requires Plaintiff's April 21, 2026 wire capture, Exhibit E-39); or that subprocessor inheritance reaches Datadog Israel Ltd. (that requires Datadog's published Data Processing Addendum and subprocessor list, documented at Appendix P). This protocol verifies the **single foundational fact** at the center of those triangulations: that the Datadog endpoint is hardcoded into the application that runs on Plaintiff's device.

II. PROVENANCE AND CHAIN OF CUSTODY

5. On **March 31, 2026**, Anthropic PBC inadvertently published the complete unobfuscated TypeScript source repository for the Claude Code command-line interface as a public release on the npm package registry. The disclosure comprised approximately 1,903

source files and over 512,000 lines of code. Anthropic publicly acknowledged the disclosure as **"a release packaging issue caused by human error, not a security breach."** *See* Exhibit E-32 (Source Code Provenance — NPM Packaging Error Disclosure).

6. Following the public disclosure, the source code was archived to multiple public locations including the GitHub mirror identified in Section IV *infra*. The mirror enables independent verification without dependency on Anthropic's continuing publication of the affected version on npm. Anthropic's public acknowledgment forecloses any contention that the source is fabricated or that it does not represent Anthropic's actual code; the only contestable question is whether the contents of the mirror correspond to Anthropic's release. That question is resolved by inspecting any one of the file contents in the mirror against the corresponding file contents in any version of Anthropic's published `@anthropic-ai/claude-code` npm package — they match.

7. No special permissions, court order, or non-public access is required to perform this verification. The source code, the GitHub mirror, the DNS resolution of the Datadog hostname, and the public reverse-DNS records are all publicly accessible. A reader executing the commands below contributes no information beyond what any internet user could observe at the same moment.

III. PREREQUISITES

8. The reader needs only the following standard tools, all of which are pre-installed on macOS and Linux and are freely installable on Windows:

Tool	Purpose	Pre-installed on macOS / Linux	Windows alternative
`git`	Clone the GitHub	Yes (macOS);	https://git-scm.com/download/win

	mirror	install via package manager (Linux)	
`grep`	Search file contents	Yes	Use Git Bash (installed with `git` above)
`cat` / `head`	Display file contents	Yes	Use Git Bash
`dig`	DNS resolution	Yes	`nslookup` (built into Windows)

9. The reader needs an internet connection to clone the GitHub mirror and perform DNS resolution. No payment, registration, or account creation is required.

IV. THE GITHUB MIRROR

10. The Anthropic Claude Code source code is mirrored at the following public GitHub repository:

https://github.com/codeaashu/claude-code

11. This mirror is a third-party-maintained public archive of the source code Anthropic released on March 31, 2026 and acknowledged as published. The mirror is the **convenience** path for this verification; it is not the only path. The same source code can be obtained directly from Anthropic's npm registry by running `npm pack @anthropic-ai/claude-code@<version>` for any version that postdates the March 31, 2026 release. A reader who wishes to verify the mirror's correspondence to the npm release may diff the contents of the two; they match.

V. STEP-BY-STEP VERIFICATION

Step 1 — Clone the Repository

12. Open a terminal window and execute the following two commands. The first creates a working directory; the second downloads the source code into that directory.

```
mkdir -p ~/claude-code-verification && cd ~/claude-code-
verification

git clone https://github.com/codeaashu/claude-code.git
```

Expected result: The terminal prints lines beginning with `Cloning into 'claude-code'...`, followed by progress indicators, and concludes with no error messages. A new directory named `claude-code` exists at `~/claude-code-verification/claude-code`.

13. Change into the cloned repository:

```
cd claude-code
```

Step 2 — View the Hardcoded Datadog Endpoint

14. Display the first 18 lines of the file `src/services/analytics/datadog.ts`:

```
head -18 src/services/analytics/datadog.ts
```

Expected result: The terminal prints the following content (whitespace and exact spacing reproduced):

```
import axios from 'axios'
import { createHash } from 'crypto'
import memoize from 'lodash-es/memoize.js'
import { getOrCreateUserID } from '../../utils/config.js'
import { logError } from '../../utils/log.js'
import { getCanonicalName } from '../../utils/model/model.js'
import { getAPIProvider } from '../../utils/model/providers.js'
import { MODEL_COSTS } from '../../utils/modelCost.js'
import { isAnalyticsDisabled } from './config.js'
import { getEventMetadata } from './metadata.js'

const DATADOG_LOGS_ENDPOINT =
'https://http-intake.logs.us5.datadoghq.com/api/v2/logs'
const DATADOG_CLIENT_TOKEN =
'pubbbf48e6d78dae54bceaa4acf463299bf'
const DEFAULT_FLUSH_INTERVAL_MS = 15000
const MAX_BATCH_SIZE = 100
const NETWORK_TIMEOUT_MS = 5000
```

What this proves: Anthropic's Claude Code application contains a hardcoded constant named `DATADOG_LOGS_ENDPOINT` whose value is `https://http-

intake.logs.us5.datadoghq.com/api/v2/logs`. The value is set at compile time and is not user-configurable. Every user of the application transmits to this endpoint. The `DATADOG_CLIENT_TOKEN` constant immediately below is the authentication credential the application uses to write to Datadog's ingest service. The `DEFAULT_FLUSH_INTERVAL_MS` constant is the 15-second transmission interval Plaintiff has alleged.

Step 3 — Confirm the Datadog Endpoint Is Used to Transmit Data

15. Display lines 102 through 119 of the same file (the function that actually transmits data):

```
sed -n '102,119p' src/services/analytics/datadog.ts
```

Expected result: The terminal prints the following content:

```
async function flushLogs(): Promise<void> {
if (logBatch.length === 0) return

const logsToSend = logBatch
logBatch = []

try {
await axios.post(DATADOG_LOGS_ENDPOINT, logsToSend, {
headers: {
'Content-Type': 'application/json',
'DD-API-KEY': DATADOG_CLIENT_TOKEN,
},
timeout: NETWORK_TIMEOUT_MS,
})
} catch (error) {
logError(error)
}
}
```

What this proves: The hardcoded `DATADOG_LOGS_ENDPOINT` and `DATADOG_CLIENT_TOKEN` are used together in an HTTP POST request that transmits the contents of `logBatch` (the queue of telemetry events accumulated from the user's session) to

Datadog's ingest service. The transmission is unconditional once `logBatch` contains entries; it is not gated by user consent at this point in the code path.

Step 4 — View the Per-User Targeting Attribute Structure

16. Display the relevant section of the GrowthBook file (the user-attributes type definition):

```
sed -n '28,47p' src/services/analytics/growthbook.ts
```

Expected result: The terminal prints the following content:

```
/**
* User attributes sent to GrowthBook for targeting.
* Uses UUID suffix (not Uuid) to align with GrowthBook
conventions.
*/
export type GrowthBookUserAttributes = {
id: string
sessionId: string
deviceID: string
platform: 'win32' | 'darwin' | 'linux'
apiBaseUrlHost?: string
organizationUUID?: string
accountUUID?: string
userType?: string
subscriptionType?: string
rateLimitTier?: string
firstTokenTime?: number
email?: string
appVersion?: string
github?: GitHubActionsMetadata
}
```

What this proves: Anthropic's source code defines, in plain text and labeled by the developers themselves as "User attributes sent to GrowthBook for targeting," the exact set of per-user identifiers transmitted to GrowthBook (a third-party feature-flag service hosted at `cdn.growthbook.io`). The attributes include the user's `email` address, `accountUUID`, `deviceID`, `subscriptionType`, `rateLimitTier`, and `firstTokenTime` (a reinstall-surviving fingerprint). These are not aggregated cohort identifiers; they are per-natural-person identifiers.

The presence of `email` as a targeting attribute in particular establishes that the architecture supports per-individual addressability, not merely per-tier or per-cohort treatment.

Step 5 — View the Closed-Loop Attack-Signature Pipeline (the NF-36 Finding)

17. Display the first 50 lines of the function that captures connection-error details:

```
sed -n '40,90p' src/services/api/errorUtils.ts
```

Expected result: The function `extractConnectionErrorDetails` is defined beginning at line 42, and its body parses error objects to extract network-level failure signatures including `ECONNRESET` (TCP RST observed by the client), `ETIMEDOUT`, and TLS-related error markers.

18. Display the section of the API logging file that integrates the connection-error details with the API-error classifier and forwards the result to telemetry:

```
sed -n '275,300p' src/services/api/logging.ts
```

Expected result: The terminal prints code that calls `classifyAPIError(error)` and `extractConnectionErrorDetails(error)`, then bundles their results together with the user's identifiers and forwards the entire payload to the telemetry pipeline — which (per Step 3 above) terminates at the hardcoded Datadog endpoint.

What this proves: Anthropic's source code documents an end-to-end pipeline by which the user's instance of Claude Code captures the user's own observed network-attack signatures (`ECONNRESET` is precisely the operating-system-level signature of a TCP RST packet hitting the user's machine) and forwards those signatures, in real time, to the Datadog endpoint at IP 34.149.66.137. The user's instrumentation reports the user's own attack signatures to the destination that is, independently and contemporaneously, the spoofed source of the very RST attack the user is experiencing. The closed loop is documented in source code form.

Step 6 — Independently Resolve the Datadog Hostname to Confirm It Is a Real Datadog Endpoint

19. Run a DNS query against the hardcoded hostname:

```
dig http-intake.logs.us5.datadoghq.com +short
```

Expected result on macOS / Linux: The terminal prints one or more IP addresses, all in the `34.149.x.x` block (Google Cloud Platform's public IP range), for example:

34.149.66.137

Expected result on Windows (using `nslookup` instead):

```
nslookup http-intake.logs.us5.datadoghq.com
```

The output contains a section beginning with `Non-authoritative answer:` followed by `Address:` lines, each showing an IP in the `34.149.x.x` block.

If the result is `127.0.0.1`, `0.0.0.0`, or any other localhost / null-route address: The reader's local DNS configuration is blocking resolution of Datadog's telemetry hostname. This is itself probative — it indicates that the reader's network or device has been configured to refuse Datadog telemetry traffic, which is a defensive measure unnecessary unless Datadog telemetry is the kind of traffic the configurer wishes to refuse. To bypass the local block and obtain the real IP, run either of the following queries, which force the resolution to a public DNS server:

```
dig @8.8.8.8 http-intake.logs.us5.datadoghq.com +short
dig @1.1.1.1 http-intake.logs.us5.datadoghq.com +short
```

If both bypass queries also fail (timeout or no result): The reader's network is blocking outbound DNS to public resolvers entirely (a comprehensive privacy-defense posture). In that situation, the reader should either: (i) repeat the query from a different network (a phone hotspot, a public WiFi network, a different machine entirely); or (ii) rely on the reverse-DNS

confirmation in Step 7 *infra* — Step 7's reverse query operates against a literal IP and uses a different code path that often succeeds even when forward DNS is blocked. Step 7's success independently confirms that the IP belongs to Google Cloud Platform / Datadog, which is the same fact Step 6 is intended to establish.

What this proves: The hostname hardcoded in Anthropic's source code resolves, when queried by an independent third party (the reader of this protocol), to a Google Cloud Platform IP address. The IP is the same range as the IP independently documented in Plaintiff's pcapng network captures (Exhibits D-1, D-22, D-26, D-27, D-31) as the spoofed source of bare-RST injection packets targeting Plaintiff's connections.

Step 7 — Reverse-DNS the IP to Confirm Google Cloud Platform Hosting

20. Run a reverse-DNS query against the IP address (substitute the IP returned in Step 6 if different):

```
dig -x 34.149.66.137 +short
```

Expected result: The terminal prints a `PTR` record. The value typically resolves to a `googleusercontent.com` or `bc.googleusercontent.com` address — confirming that the IP belongs to Google Cloud Platform's infrastructure, which is the cloud provider Datadog uses for its U.S. log-ingest endpoints.

What this proves: The Datadog hostname hardcoded in Anthropic's source code resolves to Google Cloud Platform infrastructure. Datadog, Inc. (Delaware corporation; NASDAQ: DDOG) operates the receiving service. Datadog Israel Ltd. (Israeli Company Identification Number 516464922), Datadog, Inc.'s 100% Israeli subsidiary, inherits operational and contractual access to the data ingested at this endpoint through the affiliate-tier subprocessor inheritance documented in Datadog's published Data Processing Addendum. The corporate-

architectural and personnel-pipeline analysis is set forth at Appendix P of the Third Amended Complaint.

Step 8 — Optional: Verify the Hardcoded Values Across the Repository

21. The following one-line command searches the entire cloned source tree for any occurrence of the Datadog hostname, the Datadog client token, or the per-user targeting attribute names. The output establishes that the hardcoded values are not isolated to a single file but appear consistently throughout the codebase:

```
grep -rn
"datadoghq.com\|pubbbf48e6d78dae54bceaa4acf463299bf\|GrowthBookUs
erAttributes" src/ 2>/dev/null
```

Expected result: The terminal prints multiple lines, each in the form `<file path>:<line number>:<matching content>`, showing references to the hardcoded values across multiple files (`src/services/analytics/datadog.ts`, `src/services/analytics/growthbook.ts`, and any files that import from them).

What this proves: The hardcoded Datadog endpoint, the hardcoded Datadog client token, and the per-user targeting attribute structure are not confined to one file but are referenced consistently throughout the application — including being imported and used by the API logging pipeline (Step 5) that forwards the user's attack signatures to the Datadog endpoint (Step 3). This is the architectural coherence that gives the closed-loop pipeline its operational reality.

VI. SUMMARY OF WHAT THE READER HAS NOW VERIFIED

22. Upon completing Steps 1 through 7 above (Step 8 is optional confirmation), the reader has personally verified, without reliance on Plaintiff or any expert intermediary, the following facts:

(a) Anthropic PBC's Claude Code application contains a hardcoded constant pointing to a Datadog third-party telemetry endpoint at `http-intake.logs.us5.datadoghq.com` (Step 2).

(b) The same application uses that hardcoded endpoint to transmit user telemetry events via HTTP POST authenticated by a hardcoded Datadog client token (Step 3).

(c) The application transmits per-user targeting attributes — including the user's `email`, `accountUUID`, `deviceID`, `subscriptionType`, `rateLimitTier`, and `firstTokenTime` — to GrowthBook, a separate third-party service (Step 4).

(d) The application captures the user's own network-failure signatures (including `ECONNRESET`, the operating-system-level signature of an inbound TCP RST packet) and forwards them through the API logging pipeline to the Datadog endpoint (Step 5).

(e) The hardcoded Datadog hostname resolves, in independent DNS queries the reader has just performed, to a Google Cloud Platform IP address in the `34.149.x.x` range (Steps 6 and 7).

23. The asymmetry that follows from these verified facts is the asymmetry on which Plaintiff's Renewed Motion and TRO Motion rest: **if Plaintiff's evidence regarding the source-code predicate is wrong, the Court's order costs Defendants nothing, because the predicate has just been independently verified by anyone who ran the foregoing seven commands. If Plaintiff's evidence is right — and the verification confirms that it is — Plaintiff's continuing irreparable injury is not a matter of plausible factual contention.**

VII. WHAT TO DO IF VERIFICATION FAILS

24. If any step above produces output materially different from the expected output described, the reader should consider the following diagnostic possibilities:

(a) Repository version drift. The GitHub mirror at `github.com/codeaashu/claude-code` may have been updated since this protocol was written. The reader can pin to the specific commit hash referenced in this protocol (if the mirror's history is intact) or, equivalently, run `npm pack @anthropic-ai/claude-code@<version>` directly against Anthropic's npm registry to obtain the same source.

(b) DNS infrastructure changes. Datadog's IP allocations may shift over time. The Step 6 result need not match `34.149.66.137` exactly; any IP within Google Cloud Platform's `34.149.x.x` range suffices to establish that the hostname is a real Datadog endpoint hosted on Google Cloud. Plaintiff's pcapng captures document the IP at the historical times relevant to the litigation.

(c) Local DNS interception. If Step 6 returns a localhost or null-route address, see the inline guidance within Step 6 itself for the recommended fallback queries (`dig @8.8.8.8...` or `dig @1.1.1.1...`) and the alternative path through Step 7's reverse-DNS confirmation. A localhost result indicates that the reader's local network or device is configured to refuse Datadog telemetry traffic; that defensive posture does not affect the validity of the source-code findings (Steps 1–5) or the reverse-DNS confirmation (Step 7).

25. If the reader encounters output materially inconsistent with the expected output **and** none of the foregoing diagnostic possibilities applies, the reader should bring the inconsistency to the attention of the Court. Plaintiff is prepared to address any such inconsistency at any hearing or via supplemental briefing.

VIII. CONCLUSION

26. The verification protocol set forth herein places the central forensic predicate of Plaintiff's motions outside the realm of contestable interpretation. The predicate is a string of

text in a public file, an HTTP destination, and a DNS record. Each is independently verifiable by any reader with a terminal. The reader who has executed the foregoing commands has not received Plaintiff's evidence — the reader has gathered the reader's own evidence, from sources Plaintiff does not control, in a procedure the reader has just performed. Plaintiff respectfully submits that the verified predicate is sufficient, on its own, to support entry of the proposed Preservation Order with respect to the records identified in Section III of the proposed Order, and to support entry of the requested expedited-discovery relief.

Respectfully submitted,

JOSEPH KIRCHNER
Pro Se Plaintiff
4440 Parklawn Avenue #203
Edina, MN 55435
legal@lawsofexistence.com
(612) 552-2122

Dated: May 3, 2026