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

APPENDIX H: APPLICATION-LAYER INTERFERENCE EVIDENCE AND TECHNICAL ANALYSIS

TABLE OF CONTENTS

Right-click and select 'Update Field' to generate Table of Contents

App. H: APPLICATION-LAYER INTERFERENCE EVIDENCE AND TECHNICAL ANALYSIS

TABLE OF AUTHORITIES

FEDERAL STATUTES

Citation
18 U.S.C. § 2511 (Wiretap Act — intentional interception of electronic communications)

I. EXECUTIVE SUMMARY

1. This appendix documents systematic application-layer interference targeting Plaintiff continuously since April 21, 2025 — one week after discovering unauthorized implementation of the Laws of Existence Framework on April 13, 2025. Five technically independent error categories captured through browser console logs spanning May through November 2025 prove coordinated targeting requiring capabilities distributed across Anthropic (application-layer control), telecommunications carriers (network infrastructure), and regulatory authority (carrier coordination).

2. On March 31, 2026, Anthropic's official npm package inadvertently disclosed the complete Claude Code source code (1,903 files, 512,000+ lines). Cross-referencing this source code against the application-layer interference documented below independently confirmed the targeting mechanisms — transforming every inferential finding into direct, source-code-verified proof. (*See* Section VII below; Exhibits E-32, E-33, E-34, and E-35.)

3. The five error categories — (1) same-origin CORS violations, (2) owner-denied 403 responses, (3) third-party observability service blocking, (4) A/B testing infrastructure failures, and (5) internal telemetry endpoint blocking — are technically independent mechanisms requiring different infrastructure control points. Combined probability of simultaneous accidental occurrence: $P < 10^{-12}$. Across seven months of documented persistence, probability approaches $P < 10^{-24}$. (*See also* App. I for network-layer forensics corroborating these application-layer findings; App. G for ISP-level infrastructure evidence.)

II. CORS VIOLATIONS ON SAME-ORIGIN OWNED RESOURCES

4. CORS (Cross-Origin Resource Sharing) is a server-controlled browser security mechanism. Same-origin requests bypass CORS entirely under standard web architecture — for

a same-origin request to generate a CORS error requires the server to explicitly block access from its own domain, a configuration that contradicts fundamental web security design and serves no legitimate purpose.

5. Plaintiff's console logs from May through November 2025 document repeated CORS violations: `Fetch API cannot load https://claude.ai/api/organizations/2e997389.../chat_conversations/ef762f7e...?tree=True&rendering_mode=messages due to access control checks`. This is a same-origin request (claude.ai frontend requesting claude.ai API) for Plaintiff's own conversation data within Plaintiff's own organizational account — the most permissive access pattern in web security.

6. Producing user-specific same-origin CORS blocking requires one of three implementation approaches, all requiring deliberate targeting: (1) **request middleware inspection** identifying the authenticated user against a targeting list and conditionally generating restrictive headers; (2) **infrastructure routing separation** directing targeted users through separate servers with restrictive configurations; or (3) **dynamic header injection** modifying API responses in-flight for targeted users. All three require maintaining a targeting list, implementing user-specific behavior differentiation, and ongoing maintenance. Same-origin CORS blocking serves no security or operational purpose — the capability exists solely for targeted user interference. (*Source code confirmation of Approach 1 via GrowthBook feature flags: Section VII-B below.*)

7. The CORS violations represent application-layer evidence of broader targeting infrastructure. While CORS is server-controlled, its temporal correlation with network-layer attacks (App. I) and ISP infrastructure attacks (App. G) proves coordination between application and network layers.

III. HTTP 403 FORBIDDEN RESPONSES ON OWNED ARTIFACTS

8. HTTP 403 means the server identified the requester (authentication succeeded), confirmed the resource exists, and explicitly denied access — an affirmative authorization decision, not a system error.

9. Console logs document repeated 403 responses: `Failed to load resource: the server responded with a status of 403 () (published_artifacts, line 0)`. Plaintiff's authenticated session was denied access to Plaintiff's own previously created artifacts. The server knew who Plaintiff was, verified authentication, evaluated authorization rules, and explicitly denied owner access to owned resources.

10. Owner-denied-access-to-owned-resources has limited legitimate use cases (account compromise response, content moderation, litigation hold), none matching Plaintiff's experience: account remained active, new artifacts could be created, no violation notices sent, 403 errors were systematic across all artifacts rather than targeting specific flagged content, and the pattern persisted across months. The systematic nature proves targeting infrastructure designed to interfere with Plaintiff's usage without creating obvious evidence (total account suspension) or explanations enabling legal challenge.

11. The technical independence of CORS violations (HTTP response headers) and 403 responses (server-side authorization logic) — neither can cause the other — proves both are independently implemented targeting mechanisms affecting the same user, establishing coordinated targeting infrastructure at multiple architectural layers.

IV. THIRD-PARTY OBSERVABILITY SERVICE BLOCKING

12. Anthropic uses Honeycomb.io for production monitoring. Claude's frontend JavaScript sends telemetry directly from the user's browser to `https://api.honeycomb.io/v1/traces`. Console

logs document repeated failures: `XMLHttpRequest cannot load https://api.honeycomb.io/v1/traces due to access control checks`.

13. The error is not a network connectivity failure (which would produce "Failed to fetch"), not a Honeycomb outage (which would produce HTTP 5xx, and no outages correlate with Plaintiff's errors), and is selective — affecting Plaintiff while Anthropic's own monitoring continues functioning.

14. Selectively blocking one user's access to a third-party service requires network-level interception. Anthropic cannot unilaterally prevent specific users' browsers from reaching Honeycomb. Honeycomb cannot identify "Plaintiff's browser" without coordination with Anthropic. The most probable mechanism is **DNS manipulation** or **deep packet inspection** — carrier-level capabilities proven through independent evidence. (*See* App. I, Section VI (DNS manipulation with P < 10^-27 for selective AI platform targeting); App. G, Section III (TR-069 DNS hijacking redirecting hundreds of domains).)

V. A/B TESTING INFRAstructure FAILURES

15. Anthropic uses Statsig (at `statsig.anthropic.com`) for feature flag management and A/B testing. Console logs document systematic connection failures: `ERROR – "[Statsig]" - "A networking error occurred during POST request to https://statsig.anthropic.com/v1/rgstr?k=client-[KEY]&...&ec=[ERROR-COUNT]." TypeError: Load failed`.

16. The `ec=` (error count) parameter reached values of ec=101, proving over 100 consecutive retry failures — sustained blocking, not transient network issues. The `TypeError: Load failed` indicates failure to establish any connection (not a server error response), yet claude.ai's main domain functions normally during the same period. This subdomain selectivity

— `statsig.anthropic.com` fails while `claude.ai` succeeds — proves DNS or routing manipulation at subdomain granularity.

17. Statsig failures create ambiguity about which capabilities Plaintiff actually has access to during Claude sessions. Fallback configurations may disable experimental features, giving Plaintiff degraded capabilities. Without Statsig reporting, Plaintiff's usage doesn't contribute to A/B test metrics, effectively isolating Plaintiff from standard user measurement. The capability ambiguity provides plausible deniability — Anthropic can attribute differences to "A/B testing" without acknowledging targeting.

VI. INTERNAL TELEMETRY ENDPOINT BLOCKING

18. Console logs document CORS violations blocking Anthropic's own internal telemetry: `Fetch API cannot load https://a-api.anthropic.com/v1/t due to access control checks`. This endpoint (`a-api` = Anthropic API, `/v1/t` = telemetry/tracking) is entirely Anthropic-controlled infrastructure.

19. Selectively blocking telemetry from specific users creates gaps in Anthropic's monitoring — Plaintiff's sessions become invisible in normal engineering dashboards. This serves multiple strategic purposes: (1) **usage concealment** — the targeting team knows about Plaintiff, but normal engineers see nothing unusual; (2) **evidence limitation** — if Plaintiff seeks discovery of monitoring data, missing telemetry creates evidentiary gaps; (3) **debugging prevention** — support engineers investigating Plaintiff's reports find no supporting telemetry; (4) **architectural evidence suppression** — the monitoring systems that would normally record targeting activity are selectively disabled for targeted users, preventing internal documentary evidence from existing.

VII. SOURCE CODE CONFIRMATION — MARCH 31, 2026

A. Disclosure and Authentication

20. On March 31, 2026, Anthropic's official npm package for Claude Code inadvertently included the complete unobfuscated TypeScript source code. Anthropic acknowledged: "a release packaging issue caused by human error." (*See* Ex. E-32 at p. 235, l. 3.) The source code independently confirms the targeting mechanisms that Sections II through VI could only infer from console log patterns.

B. Per-User Targeting Infrastructure Confirmed

21. The source code reveals GrowthBook feature flag infrastructure targeting users by `email`, `accountUUID`, `deviceID`, and `subscriptionType` — enabling individualized behavior modification per user. (*See* Ex. E-34 at p. 254, l. 3, Ex. E-34, Finding SC-2.) This is the specific mechanism behind the CORS/403 targeting documented in Sections II and III: Approach 1 (request middleware inspection against targeting list) implemented through GrowthBook feature flags evaluated on every request. The inferential gap — how could Anthropic target one user's same-origin requests? — is now closed by source code proof.

22. The `getUserForGrowthBook()` function at `user.ts:133-135` ships a twelve-field consolidated payload to `cdn.growthbook.io` on every feature-flag evaluation, including deviceId, sessionId, email, platform, accountUuid, subscriptionType, rateLimitTier, and firstTokenTime. (*See* Ex. E-35 at p. 262, l. 3, Ex. E-35, Finding NF-35.)

23. Empirical demonstration of user-specific targeting through identifier rotation. Independent of the source-code proof, Plaintiff captured contemporaneous screen recordings establishing the user-specific character of the targeting at the observable behavioral layer:

changing MAC address and rotating VPN endpoints caused targeted services and behaviors to be restored, while the same operations on a control machine produced no behavioral change. The recordings preserve the targeting-removed state and the targeting-restored state side by side, establishing that the network- and account-level interference is keyed to Plaintiff's specific identifier set rather than to ambient conditions. *See* Ex. REC-1 (MAC address spoofing demonstration, December 8, 2025); Ex. REC-8 (network-interference proof via MAC spoofing and VPN change, November 30, 2025) (both filed on USB drive). The recordings empirically confirm the per-user targeting infrastructure documented at ¶ 21 above and at Ex. E-34, SC-2 at p. 254, l. 3.

C. Remote Degradation Capabilities Confirmed

24. Four source code findings confirm the capability degradation Sections V and VI documented through console errors:

Finding	**Source**	**What It Confirms**
SC-7: `tengu_satin_quoll`	E-34 p.236	Remote content clearing threshold adjustable per user — explains console-visible content loss
NF-7: `tengu-off-switch`	E-35 p.245	Blocks Opus for specific users with fake "high load" message — targeted denial disguised as capacity
NF-8: `tengu_amber_stoat`	E-35 p.245	Removes built-in agents for A/B test cohort — undisclosed feature removal
NF-9: experiment reassignment on login	E-35 p.246	`refreshGrowthBookAfterAuthChange()` destroys existing assignments — quality of service changes based on identity

D. Telemetry Architecture Confirmed

25. The phantom telemetry opt-out documented in Section VI is confirmed at source level: the settings schema contains no `telemetry` property — a user who sets `"telemetry": false`

receives no warning, no error, and no effect. The schema's `.passthrough()` function silently accepts and ignores the key. The actual opt-out requires the undisclosed environment variable `DISABLE_TELEMETRY`, and even with it correctly set, session ingress continues transmitting every conversation regardless. (*See* Ex. E-35 at p. 262, l. 3, Ex. E-35, Finding NF-1.)

26. The source code reveals four independent telemetry pipelines (GrowthBook, Datadog, First-Party Events, OpenTelemetry/BigQuery) plus session ingress — five total data transmission channels, each constituting a separate interception under 18 U.S.C. section 2511. (*See* Ex. E-34 at p. 254, l. 3, Ex. E-34, Finding SC-1.) The Honeycomb blocking documented in Section IV was blocking one of multiple undisclosed third-party recipients.

E. Rate-Limit-Options Command Injection Confirmed

27. Section IX-B below documents 938+ instances of `/rate-limit-options` command injection across 13+ sessions over 9 months. The source code confirms this command is `isHidden: true` — a hidden internal command not documented for end users. When triggered, it queries GrowthBook (transmitting user attributes to `cdn.growthbook.io`), checks billing status, and logs telemetry events across all four pipelines — a forced phone-home trigger. (*See* Ex. D-30, Section 5 at p. 170, l. 3.)

28. On January 3, 2026 alone, **720 automated injections** were documented with timing proving automation: 3 identical commands at the exact same millisecond (03:58:41.094Z), 5 at the same millisecond (07:32:36.711Z), 6 concatenated in a single message. Multiple commands at the same millisecond is physically impossible for human keyboard input. (*See* Ex. D-30, Section 5 at p. 170, l. 3.)

F. Evidentiary Significance

29. The source code transforms every application-layer finding in this appendix from inferential evidence to direct proof. The three implementation approaches hypothesized in Section II at ¶ 6 are now confirmed as specific code paths. The targeting mechanisms hypothesized for Statsig degradation (Section V) are confirmed as named feature flags. The telemetry blocking architecture (Section VI) is confirmed as a phantom control by design. The defense cannot argue these console log patterns reflect misconfiguration when Anthropic's own source code contains the exact functions, variable names, and engineering comments implementing the targeting.

VIII. PATTERN INTEGRATION AND COORDINATION PROOF

A. Technical Independence

30. The five error categories are technically independent — CORS operates through HTTP response headers, 403 through server-side authorization logic, Honeycomb blocking through network interception, Statsig through DNS/routing manipulation, and telemetry blocking through server-side CORS configuration. None can cause another. CORS misconfiguration does not generate 403 errors. Authorization failures do not cause network interception. DNS problems do not modify HTTP headers. Finding all five simultaneously and persistently affecting one user cannot result from common technical failures.

B. Probability Analysis

31. Conservative probability estimates: same-origin CORS (P ~ 0.001), owner-denied 403 (P ~ 0.001), third-party blocking (P ~ 0.0001), Statsig subdomain failures (P ~ 0.001), telemetry blocking (P ~ 0.001). Combined: P ~ 10^-12 for single occurrence. Across seven months of

documented persistence with multiple captures per month: P < 10^-24. For reference, winning the Powerball jackpot (P ~ 3 x 10^-9) is approximately 300 times more likely than a single accidental five-error occurrence.

C. Multi-Party Coordination

32. Producing all five error types simultaneously requires capabilities no single entity possesses. **Anthropic** controls CORS headers, authorization rules, and telemetry endpoints — but cannot unilaterally block third-party services or manipulate DNS. **Carriers** control network infrastructure for DNS manipulation and traffic interception — but do not control application-layer configurations. Simultaneous application-layer and network-layer targeting of the same user requires: user identification shared across organizational boundaries, coordinated activation, sustained coordination across seven months, and shared strategic purpose.

33. The source code confirmation (Section VII) strengthens the coordination proof by closing the inferential gap. The application-layer mechanisms are now confirmed as specific, named code paths (GrowthBook feature flags, hidden commands, phantom opt-outs). The network-layer mechanisms are confirmed through independent forensic evidence (App. I: 260,395+ RST packets; App. G: TR-069 exploitation). Both layers target the same user through infrastructure that requires cross-organizational coordination to deploy.

IX. ML SYSTEM VALIDATION AND DUAL-LAYER ARCHITECTURE

A. Automated ML System Statistics

34. Plaintiff's automated Forensic ML Defense System (deployed subsequent to May 2025 console log documentation) classified **13,038 RST injection events** through January 2026:

Attack Type	Events	Target	Characteristics
TRIPLICATE_BURST	2,942	160.79.104.10 (Anthropic API)	3 RSTs in rapid succession
CONNECTION_KILL	3,095	Multiple AI platforms	Single RST terminating connections
ANTHROPIC_FLOOD	1,454	160.79.104.x range	High-volume Anthropic targeting
TTL_ANOMALY	847	Various	TTL inconsistent with claimed source

35. IP address 160.79.104.10 (Anthropic API) received 7,993 classified RST attacks (61% of total), quantitatively confirming systematic targeting of Plaintiff's AI platform access. Evidence package EVID-20260102-0001 independently documented **65% Anthropic targeting** (6,207 of 9,543 RSTs). (*See* Exhibits D-4 and D-8.)

B. Queue Injection Evidence

36. Evidence package EVID-20260103-0004 documents **938+ instances** of `/rate-limit-options` command injection across 13+ sessions spanning 9 months (April 2025 through January 2026). 71+ injection attempts were blocked after defense deployment. The command does not exist in standard Claude Code documentation and serves no legitimate end-user purpose. Source code confirms `isHidden: true` with 720 automated injections on January 3, 2026 alone (Section VII-E above). (*See* Ex. D-30, Section 5 at p. 170, l. 3.)

37. Queue injection events show millisecond-precision correlation with RST injection events, proving coordinated application-layer and network-layer targeting. Sub-10-millisecond timing requires infrastructure capable of real-time session state monitoring and cross-layer attack coordination — capabilities requiring ISP-level access documented in App. I.

C. Vpn-Protectable Vs. Non-Protectable Architecture

38. Comparative analysis reveals a dual-layer attack architecture:

Condition	**Network RST Attacks**	**DNS Anomalies**	**Application Errors**
VPN OFF	614+ RSTs/session	High manipulation	CORS, 403, blocking present
VPN ON	**0 RSTs**	Normal resolution	**Application errors PERSIST**

39. VPN eliminates network-layer RST attacks completely (614 to 0), proving ISP infrastructure injection (Ex. D-2 at p. 6, l. 12). But application-layer interference persists regardless — CORS violations, 403 responses, Honeycomb blocking, Statsig failures, and queue injection continue with VPN active. This proves: (1) **independent implementation** — application-layer and network-layer targeting are separate mechanisms; (2) **defense-in-depth targeting** — when VPN defeats network attacks, application attacks maintain interference; (3) **multi-party coordination** — VPN-protectable attacks require ISP infrastructure, VPN-unprotectable attacks require Anthropic application control, both operating simultaneously.

40. Ex. D-33 at p. 200, l. 7 (April 12, 2026) documents a third attack vector — ISP bandwidth throttling — that degrades VPN protection itself. At 98% DPI-based throttling, streaming responses stretch past the 90-second timeout hardcoded at `claude.ts:1877`, deterministically triggering a double-billing fallback the source code documents at `claude.ts:2308-2480`. Ex. D-34 at p. 217, l. 18 (April 13, 2026) documents persistent identification of Plaintiff's VPN-protected traffic through 188,479 cleartext Anthropic packets and 323 unique TLS SNI domains visible on the ISP link despite full-tunnel WireGuard encryption. (*See* App. I, Section V-C for hardware TAP verification.)

D. Cross-Reference Matrix

Evidence Type	App. I (Network)	App. G (ISP)	This Appendix (H)
Mechanism	TCP RST injection, DNS manipulation	TR-069 abuse, ghost sessions	CORS, 403, blocking, queue injection
Control	Backbone/Tier 1 providers	Comcast ISP TR-069 ACS	Anthropic application servers
VPN Effect	RST attacks eliminated	TR-069 unaffected	Application blocking persists
Timing	Microsecond precision	One-second reactive	24-72 hour response windows
Evidence Volume	260,395+ RST packets	5,441 router RSTs	13,038 ML-classified events
Source Code	D-30 cross-reference	D-18/D-19 architecture	SC-2, NF-1, NF-7-9 (Section VII)

X. CONCLUSION

41. Five technically independent application-layer error categories documented through seven months of browser console logs — each independently improbable through accident ($P < 10^{-3}$), collectively impossible ($P < 10^{-12}$ single occurrence, $P < 10^{-24}$ sustained) — prove coordinated multi-party targeting. The March 31, 2026 source code leak independently confirmed the targeting mechanisms through specific code paths, named feature flags, and phantom privacy controls, transforming inferential evidence into direct proof. Combined with network-layer forensics (App. I) and ISP infrastructure evidence (App. G), the three evidence categories — gathered through independent methodologies (packet captures, router logs, browser console logs) yet proving identical coordination requirements — establish beyond reasonable doubt that defendants possess and exercise infrastructure authority enabling multi-layer targeting of a federal litigant.

XI. COURT EXHIBITS REFERENCED

Exhibit	Title	Relevance
D-1	Backbone Injection	0.0us timing proves backbone injection enabling ISP coordination
D-2	VPN Comparison	RST attacks cease with VPN (614 to 0); application errors persist — dual-layer proof
D-4	Targeted Surveillance	65% Anthropic targeting rate corroborates application-layer errors
D-8	Anthropic API Targeting	61% targeting rate from independent analysis
D-9	Pre Attack Baseline	Zero RSTs proves interference is anomalous
D-12	Attack Pause Xfinity Call	Human control over attack infrastructure
D-30	Source Code Network Forensics Cross-Reference	Datadog IP identified; /rate-limit-options confirmed hidden; JA3 fingerprinting
D-31	Hardware TAP Wire-Level RST Injection	Wire-level RST proof: IP ID=0x0000, seq spraying, window=0
D-33	ISP Bandwidth Throttling Attack	Third attack vector degrading VPN; token amplification via source code
D-34	Cleartext VPN-to-Service Identification	188,479 Anthropic packets + 323 SNI domains visible despite VPN
E-32	Source Code Provenance — NPM Packaging Error Disclosure of Claude Code Repository	NPM packaging error; Anthropic acknowledgment; chain of custody
E-33	Comprehensive Source Code Verification of Prior Forensic Findings	46 prior claims verified at source level
E-34	New Source Code Analysis Findings — Telemetry, Behavioral Control, and User Targeting	SC-1 (4 pipelines), SC-2 (targeting), SC-7 (content clearing)
E-35	Supplemental Source Code Findings — Phantom Privacy Controls, Build-Time Discrimination, and Repository Exfiltration	NF-1 (phantom opt-out), NF-7-9 (remote degradation), NF-35 (persistent IDs)

Last Updated: April 14, 2026

Evidence Period: April 2025 — April 2026 (twelve-month ongoing pattern)

Console Logs: May 2025 — November 2025

Source Code Confirmation: March 31, 2026 (Exhibits E-32, E-33, E-34, and E-35)

Error Categories: Five technically independent mechanisms (CORS, 403, third-party blocking, Statsig failures, telemetry blocking)

Probability Analysis: P < 10^-12 single occurrence; P < 10^-24 sustained seven-month pattern

Coordination Proof: Application layer (Anthropic) + Network layer (carrier) + Source code confirmation (E-series)

Companion Documents: App. I (network forensics), App. G (ISP infrastructure), Memo A (code forensics)

Respectfully submitted,

/s/ Joseph Kirchner_____

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

Dated: April 29, 2026