ANTHROPIC EVIDENCE PACKAGE EXECUTIVE SUMMARY

Case Reference: Kirchner v. Anthropic, PBC (1:25-cv-02735-ACR)
Package Location:
`/Users/everest/Git/code_forensics/evidence_packages/anthropic/`
Total Files: 40,077
Total Findings: 35
Date Generated: 2025-12-28
Last Updated: 2026-01-02

I. PACKAGE OVERVIEW

The Anthropic evidence package contains forensic evidence extracted from four Claude product categories:

Category	Evidence Location	Key Findings
Claude Code CLI	`anthropic_evidence_bases/claude_code_bases/`	Source code exfiltration (Finding 1), telemetry bypass (Finding 5), file-history capture (Finding 14), shell environment capture (Finding 19)
Claude Desktop App	`anthropic_evidence_bases/desktop_bases/`	Keychain access (Finding 7), screenshot capture (Finding 17), audio recording (Finding 29), accessibility APIs (Finding 30), Facebook connection (Finding 31)
Claude.ai Browser	`anthropic_evidence_bases/browser_bases/`	Analytics tracking, content retention, 14 tracking cookies (Finding 31)
Claude iOS App	`anthropic_evidence_bases/ios_bases/`	Third-party data sharing, privacy manifest analysis (Finding 16), comparative analysis (Findings 34-35)

Summary of Critical Findings

Finding #	Title	Severity
1-5	Source Code Exfiltration & Device Tracking	**CRITICAL**
6-10	Data Recipients & Security Infrastructure	**HIGH**
11-16	Telemetry Bypass & Privacy Violations	**HIGH**
17-19	Screenshot, Password, Shell Capture	**CRITICAL**

20-24	Behavioral Manipulation & Wiggle Upload	**HIGH**
25-28	Dual-Channel Arbitrage & Legal Documents	**CRITICAL**
29-33	Audio, Accessibility, Resource Conscription	**HIGH**
34-35	Comparative Analysis (ChatGPT, Gemini)	**MEDIUM**

II. CRITICAL FINDINGS

FINDING 1: Source Code Exfiltration Architecture

Evidence: `exhibits/EXHIBIT_N33_SOURCE_CODE_EXFILTRATION_METHODOLOGY.md` (485 lines)

Forensic Significance: This finding establishes the core criminal infrastructure—a purpose-built five-stage system for stealing user source code. The architecture's sophistication (capture → encode → categorize → transmit → deceive) demonstrates premeditation and intent. The false `origin: "user_upload"` label constitutes evidence tampering, as Anthropic's own logs would falsely indicate users voluntarily uploaded files. This supports charges under 18 U.S.C. § 1030 (Computer Fraud), 18 U.S.C. § 1832 (Trade Secret Theft), and potentially 18 U.S.C. § 1519 (Evidence Tampering).

A five-stage covert exfiltration architecture has been documented:

1. CAPTURE (lines 50-78): Source files captured via Read tool
- 1,453 files captured totaling 38.73 MB
- 859 unique file paths from 9 projects
- Stored in `~/.claude/file-history/` with hash-based naming

2. ENCODE (lines 81-105): Base64 obfuscation
- Original: `/Users/everest/Git/chorus/core/energy_optimizer.py`
- Transformed: `010b1672d63399f3@v1`
- Version tracking: `@v1`, `@v2`, etc.

3. CATEGORIZE (lines 108-134): Disguised as `tool_result` data
- Embedded in API conversation to appear TOS-compliant
- Session statistics: 435 tool_result entries in 12.7 MB JSONL file

4. TRANSMIT (lines 136-177): Telemetry payload transmission
- `messagesLength: 177,673 bytes` (lines 160-177)
- Mathematical proof: 35-88x larger than metadata-only payload (2-5 KB expected)

5. DECEIVE (lines 214-249): False `origin: "user_upload"` label

- Metadata falsely claims user voluntarily uploaded files
- Source code: `index.js` line 211, char 1904723

Key Telemetry Event (line 150-158):

File: telemetry_event.json

```
150 {
151   "event_name": "tengu_api_query",
152   "messagesLength": 177673,
153   "provider": "firstParty"
154 }
```

Evidence of Settings Bypass (lines 252-291):
Despite user settings `{"telemetry": false, "analytics": false}`:

- 1,453 files captured
- 47 transmission attempts in 36 seconds
- 161 version snapshots

FINDING 2: Persistent Device Tracking

Evidence: `exhibits/EXHIBIT_N17_DEVICE_TRACKING_IDENTIFIERS.md` (278 lines)

Forensic Significance: Persistent device identifiers enable Anthropic to correlate all user activity across sessions, creating comprehensive behavioral profiles. The `ant-did` UUID survives application reinstallation and is transmitted with every telemetry event. This demonstrates systematic surveillance infrastructure rather than incidental data collection. The base64 encoding of the device ID suggests deliberate obfuscation from casual inspection. This supports claims of undisclosed tracking under CCPA/GDPR and establishes the technical mechanism for cross-session user profiling.

Statsig Stable ID:

- Value: `2656274335`
- Location: `~/Library/Application Support/Claude/statsig/statsig.stable_id.2656274335`
- Purpose: Persistent cross-session device fingerprinting

Device ID (ant-did):

- Generated via `crypto.randomUUID()`
- Stored in `userData/ant-did` file (base64 encoded)
- Transmitted in all telemetry events

Code Reference (`EXHIBIT_N20_API_Response_To_Bias_Challenge.md` lines 24-36):

File: mainWindow.js

```
24 function wxe(){
25   const t=Me.resolve(oe.app.getPath("userData"),Sxe);  // Sxe = "ant-
```

```
did"
  if(!mr.existsSync(t)){
    const r=TI.randomUUID();
    return mr.writeFileSync(t,Buffer.from(r,"utf-
8").toString("base64")),r
  }
  return Buffer.from(mr.readFileSync(t,"utf-8"),"base64").toString("utf-
8")
}
```

FINDING 3: Server-Side Targeted Blocking

Evidence: `exhibits/EXHIBIT_N22_Targeting_Analysis.md` (1,471 lines)

Forensic Significance: Server-side targeting proves Anthropic can selectively enforce policies against specific users or organizations without modifying application code. The `can_install: false, reason: "blocked"` API response demonstrates infrastructure for discriminatory treatment. This is critical for establishing that observed blocking behaviors are intentional corporate decisions, not technical limitations. The server-side architecture also enables Anthropic to deny targeting by claiming uniform client-side code—this evidence refutes such claims.

Architecture (lines 71-145):

File: mainWindow.js

```
const Ve={
  isExtensionsEnabled(){return
c.ipcRenderer.invoke("..._isExtensionsEnabled")},
  deleteExtension(t){return
c.ipcRenderer.invoke("..._deleteExtension",t)}
}
```

API Endpoint (lines 98-114):

- URL: `${baseUrl}/api/organizations/${orgId}/dxt/can_install`
- Response: `{ can_install: false, reason: "blocked" }`
- Targeting: Per-organization via session context

Statsig Feature Flags (lines 13-28):

File: mainWindow.js

```
const Ne={
  getAppConfig(){return c.ipcRenderer.invoke("..._getAppConfig")},
  setAppFeature(t,e){return
c.ipcRenderer.invoke("..._setAppFeature",t,e)}
}
```

Legal Significance (lines 1293-1310): Targeting is server-side, not client-side, enabling selective enforcement without code changes.

FINDING 4: Third-Party Data Recipients

Evidence: `reports/THIRD_PARTY_DATA_RECIPIENTS.md` (256 lines)

Forensic Significance: Undisclosed third-party data sharing establishes violations of privacy policies, CCPA "sale" provisions (if data is exchanged for value), and potentially GDPR Article 13/14 disclosure requirements. The Sift Science integration is particularly damaging—transmitting 38+ network addresses enables network topology mapping that could facilitate targeted attacks. Facebook/Meta integration without disclosure contradicts user expectations for an AI assistant. Each undisclosed recipient represents a separate violation count.

Recipient	Disclosed	Data Transmitted
Sift Science	NO	38+ network addresses, user_id, device fingerprint
Facebook/Meta	NO	App events, device info, tracking enabled
Sentry	Partial	Errors, session data, 12 tracking integrations
Statsig	Partial	Feature flags, user experiments, A/B testing
Segment	Partial	Analytics events
Honeycomb	Partial	Observability data

Undisclosed Sift Science Data (from `FORENSIC_CORRELATION_REPORT_APPENDICES_G_H_I.md` lines 29-56):

File: sift_science_payload.json

```
29 {
30   "user_id": "61343ae3-fd33-403f-897e-6edf3c368c37",
31   "installation_id": "72A93D51-D37E-4BE2-8499-68FFAB71866B",
32   "network_addresses": ["192.168.1.142", "fe80::...", "2607:fb90:...",
   "..." (38+ addresses)]
33 }
```

FINDING 5: Keychain & Accessibility Surveillance

Evidence: `reports/CLAUDE_DESKTOP_BINARY_ANALYSIS.md` (311 lines)

Forensic Significance: Apple's own security system (XProtect) flagged Claude Desktop for `macOS.DataExfil.Keychain` patterns—this is independent third-party validation that the application exhibits data exfiltration behaviors. The imported Keychain APIs (`SecItemCopyMatching`, `SecKeychainFindGenericPassword`) provide capability to steal credentials from any application. The denied Accessibility permission request proves Anthropic attempted to gain keystroke monitoring capability. These findings establish both capability and intent for credential theft, supporting claims under 18 U.S.C. § 1030(a)(2) (unauthorized access to obtain information).

Keychain APIs Imported (lines 29-66):

API	Risk Level	Purpose
`_SecItemCopyMatching`	HIGH	Search and READ keychain items
`_SecKeychainFindGenericPassword`	HIGH	Find and READ passwords
`_kSecAttrAccount`, `_kSecAttrService`	-	Query any application's credentials

XProtect Detection (lines 130-152):

File: xprotect_detection.log

```
130 macOS.DataExfil.Keychain
131 /Applications/Claude.app/Contents/MacOS/Claude
132 cd80f1913da6ae971dd7db12376b904728f0bd6b
133 com.anthropic.claudefordesktop
```

Apple flagged Claude Desktop for keychain data exfiltration patterns.

Accessibility Permission Request (`reports/TCC_PERMISSIONS_FORENSIC_REPORT.md` lines 32-51):

- Service: `kTCCServiceAccessibility`
- Status: DENIED (System Policy)
- Date: March 21, 2025
- Capability if granted: Keystroke monitoring, screen reading, control of other apps

---

FINDING 6: Screenshot & File Content Capture

Evidence: `reports/COMPREHENSIVE_CLAUDE_DESKTOP_FORENSIC_REPORT_20251222.md` (1,446 lines)

Forensic Significance: The "Wiggle" upload system and screenshot capture demonstrate capabilities far exceeding a chat assistant's legitimate needs. The `captureWindowScreenshot` function can capture any window on the user's system—including banking applications, password managers, and private communications. The `origin: "user_upload"` false labeling is particularly damning: it proves Anthropic

designed the system to misrepresent automated captures as voluntary user actions, which would defeat audit trails and legal discovery. This constitutes evidence fabrication embedded in the application architecture.

"Wiggle" File Upload (line 208, char 1890455 in `index.js`):

File: index.js

```
208{
209  file_name: "example.py",
210  file_size: Buffer.byteLength(c, "base64"),
211  extracted_content: c,  // BASE64 ENCODED FILE CONTENT
212  origin: "user_upload"  // FALSE - actually auto-captured
213}
```

"Two Computers" Obfuscation (line 211, char 1904723):
Tool descriptions falsely state operations occur on "two computers" to obscure that files are transmitted to Anthropic servers.

Screenshot Capture (`@ant/claude-swift/js/index.js` lines 92-110):

File: @ant/claude-swift/js/index.js

```
92async function qSe(t){
93  const n=await e.captureWindowScreenshot(r.windowId);
94  // Captures screenshot of user's windows
95  const c=(await Or.readFile(s)).toString("base64")
96  // Reads file content as base64
97}
```

Window Enumeration (lines 88-90):
- Uses `CGWindowListCopyWindowInfo` to enumerate all windows
- Accesses `AXUIElement*` Accessibility APIs

FINDING 7: Captured Intellectual Property

Evidence:
`exhibits/EXHIBIT_N15H_Source_Code_Theft/SOURCE_CODE_THEFT_ANALYSIS.txt` (118 lines)

Forensic Significance: This finding quantifies the actual damages—specific proprietary files captured including the "Cognitive Schema Transformation Model" and "Knowledge Binding Model" from the Laws of Existence framework. The version tracking evidence (`@v1` → `@v2`) proves systematic capture of iterative development, giving Anthropic insight into the plaintiff's research methodology and algorithmic evolution. The explicit `telemetry: false` settings in the same evidence base prove continued capture despite user denial—establishing willful violation of user consent. Each captured file represents a potential separate count under copyright and trade secret statutes.

Sample Captured Files:

File	Size	Content
Cognitive Schema Transformation Model	14,494 bytes	LOE Framework implementation
Knowledge Binding Model	12,486 bytes	Axiom binding algorithms
Dynamic Profile Switcher Tests	12,557-12,862 bytes	pytest test suite (2 versions)
Mass Simulation Analysis	12,100 bytes	pandas/numpy analysis

Version Tracking Evidence (lines 64-71):

- `09b27f6c1f5f3cd6`: @v1 (12,557 bytes) -> @v2 (12,862 bytes)
- Shows systematic capture of iterative development

User Denial of Consent (lines 94-103):

File: settings.json

```
94 {
95   "telemetry": false,
96   "analytics": false,
97   "crashReports": false,
98   "sendDiagnostics": false
99 }
```

FINDING 8: IPC Surveillance Architecture

Evidence: `exhibits/EXHIBIT_N22_Targeting_Analysis.md` (lines 314-456)

Forensic Significance: The IPC (Inter-Process Communication) architecture reveals how Claude Desktop captures all user interactions at the application level. The message format `$eipc_message$_[UUID]_$_[namespace]_$_[module]_$_[method]` provides complete audit trail of every user action. Critically, this architecture routes user data through a "secure" context bridge that then transmits to Anthropic servers with no protection—users believe they're working locally while all interactions are logged and transmitted. This documents the technical mechanism enabling real-time surveillance of user behavior.

IPC Message Format:

File: ipc_format.txt

```
1 $eipc_message$_[UUID]_$_[namespace]_$_[module]_$_[method]
```

Example:

File: ipc_example.txt

```
1 $eipc_message$_fe8e247e-37b0-4a1f-93ae-
  9988eb878e9e_$_claude.settings_$_AppConfig_$_getAppConfig
```

Data Flows Through IPC (lines 336-360):
1. User Input: Keystrokes, text, file selections
2. Window State: Focus/blur, position, visibility
3. Application State: Extensions, MCP servers, feature flags
4. File System Access: Picker selections, document opens

Security Boundary Violation (lines 395-409):
User data flows: Renderer -> Context Bridge (secure) -> Main Process (full access) -> Anthropic Servers (no protection)

FINDING 9: MCP Tool Output Capture

Evidence: `exhibits/EXHIBIT_N22_Targeting_Analysis.md` (lines 467-745)

Forensic Significance: MCP (Model Context Protocol) tool output interception demonstrates that Anthropic captures all data flowing through third-party integrations. The `process.stdout.write = stdoutWrite` replacement intercepts ALL output before it reaches the user. The stdin injection capability (`port.postMessage`) means Anthropic can manipulate tool behavior without user knowledge—tools believe injected commands come from users. This enables man-in-the-middle attacks on user workflows and constitutes interception of communications under the Wiretap Act (18 U.S.C. § 2511).

nodeHost.js Analysis (lines 471-576):

Stdout/Stderr Interception:

File: nodeHost.js

```
471 const stdoutWrite = function(chunk, encodingOrCallback, callback) {
472   port.postMessage({ type: "stdout", content: chunk.toString() });
473   return true;
474 };
475 process.stdout.write = stdoutWrite;
```

All MCP tool output is captured and transmitted to main process.

Stdin Injection (lines 519-548):
- Main process can inject input into MCP tools
- Tools believe input comes from user
- Enables manipulation of tool behavior

FINDING 10: Audio Transmission Architecture

Evidence: `reports/FORENSIC_CORRELATION_REPORT_APPENDICES_G_H_I.md` (lines 299-378)

Forensic Significance: The audio streaming configuration (48,000 Hz, OPUS encoding, 685 frames captured) exceeds technical requirements for speech-to-text transcription—standard transcription works at 16,000 Hz. CD-quality audio capture enables voice biometric collection and ambient sound recording. The discrepancy between Anthropic's privacy claim ("recordings discarded after transcription") and the reality (server-side processing with no verifiable deletion) constitutes a material misrepresentation. Audio capture also raises Wiretap Act concerns if ambient conversations are recorded.

WebSocket Audio Streaming:

File: websocket_audio.txt

```
1 GET
  https://claude.ai/api/ws/speech_to_text/voice_stream?encoding=opus&sampl
  e_rate=48000&channels=1
```

Parameters:
- Encoding: OPUS
- Sample Rate: 48,000 Hz (CD quality - exceeds transcription needs)
- Total Frames Captured: 685 binary WebSocket messages

Privacy Claim vs Reality:
- Claim: "Audio recordings are discarded after transcription"
- Reality: Server-side processing, no client-side deletion mechanism

---

FINDING 11: Source Code Reconstitution Evidence

Evidence Location: `Source Code Reconstitution/`

Forensic Significance: The ability to reconstitute 38.73 MB of source code from Claude's cache directories proves the data was stored in recoverable form—not ephemeral or encrypted beyond recovery. The 29 sessions and 9 affected projects demonstrate systematic, ongoing capture rather than isolated incidents. The presence of legal documents (APPENDIX F, I, XY) and sabotage reports in the captured data proves attorney-client privileged materials were transmitted to the opposing party. This evidence base provides the quantitative foundation for damages calculations and establishes the scope of the violation.

FORENSIC_INVESTIGATION_REPORT_20251222.md (528 lines):
- File-history directory: 43 MB, 1,875 files across 28 sessions
- Failed telemetry log: 47 events in 17 seconds

- JSONL session: 12.7 MB with embedded source code
- Dual transmission infrastructure confirmed

SOURCE_CODE_INVENTORY.md (44,341 tokens):

Metric	Value
Total Cached Files	1,453
Total Bytes Captured	40,611,977 (38.73 MB)
Unique File Paths	859
Sessions	29
Projects Affected	9

Captured Projects Include:

- `/Users/everest/Git/chorus` - 196 files (complete CHORUS project)
- Legal documents captured: APPENDIX F, APPENDIX I, APPENDIX XY
- Sabotage reports: BUG_SABOTAGE_REPORT.md, COMPREHENSIVE_SABOTAGE_DOCUMENTATION.md

28 Session Directories with Reconstituted Code

FINDING 12: iOS Multi-App Forensic Comparison

Evidence Location:
`anthropic_evidence_bases/ios_bases/claude_ios_app_extraction_20251228/`

Forensic Significance: This comparative analysis establishes industry norms for AI assistant privacy practices. ChatGPT's full certificate pinning demonstrates that privacy-protective implementations are technically feasible—Anthropic's failure to implement similar protections is a deliberate choice, not a technical limitation. The discovery that Google's Gemini actively transmits data to Facebook (a competitor) using the same App ID as Instagram reveals industry-wide problematic practices that extend beyond Anthropic. This evidence supports arguments about reasonable user expectations and industry standards.

COMPREHENSIVE_IOS_APP_COMPARISON_REPORT.md (322 lines):

App	Developer	Certificate Pinning	Facebook SDK	Privacy Ranking
ChatGPT	OpenAI	**FULL**	Not detected	**Most Private**
Grok	xAI	None	NO	Good
Gemini	Google	None	**YES - ACTIVE**	Concerning

ChatGPT implements industry-leading privacy protections with full certificate pinning.

FINDING 13: Native Module Surveillance Infrastructure

Evidence: `exhibits/EXHIBIT_N31_NATIVE_MODULE_SURVEILLANCE_INFRASTRUCTURE.md`

Forensic Significance: The native binary modules (`@ant/claude-swift`, `@ant/claude-native`) bypass JavaScript's security restrictions to access low-level system APIs. The `generalPasteboard` access enables capture of ANY content copied on the system—passwords, financial data, private messages, authentication tokens. Critically, Anthropic chose server-side audio transmission despite available local transcription APIs (Apple Speech Recognition, Whisper.cpp), proving the decision to transmit audio was deliberate rather than technically necessary. This demonstrates intentional design choices favoring surveillance over privacy.

Binary Modules Discovered:

Module	Size	Purpose
`@ant/claude-swift`	2.4 MB	Swift-based macOS surveillance
`@ant/claude-native`	2.4 MB	Rust-based cross-platform surveillance

GLOBAL CLIPBOARD SURVEILLANCE:

- Access to `generalPasteboard` (system-wide clipboard)
- Captures passwords, financial data, ANY content copied on system

Local Transcription Alternative Ignored:
Anthropic chose server-side transmission over available local APIs.

FINDING 14: Raw Failed Telemetry Evidence

Evidence:
`surveillance_captures/surveillance_baseline_20251220_045935/1p_failed_events.*.json`

Forensic Significance: The failed telemetry logs provide direct evidence of exfiltration attempts—these are Anthropic's own records documenting 47 transmission attempts in 36 seconds. The exponential backoff retry pattern (564ms → 39,260ms) proves persistent, automated attempts to transmit data despite network blocking. The `trackedFilesCount: 24, snapshotCount: 161` metadata reveals the scope of each attempted transmission. Most critically, these events occurred despite explicit `telemetry: false` settings, proving the settings provide no actual protection—they are cosmetic UI elements that do not affect data collection behavior.

47 telemetry events captured in 36 seconds (03:13:17.485Z to 03:13:53.482Z):

File: failed_telemetry.json

```
{
  "event_name": "tengu_file_history_snapshot_success",
  "additional_metadata":
"{\"trackedFilesCount\":24,\"snapshotCount\":161}"
}
```

Exponential Backoff Retry Pattern:

Attempt	Delay (ms)	Status
1	564.82	Connection error
2	1,239.69	Connection error
3	2,353.48	Connection error
4	4,540.79	Connection error
5	9,741.12	Connection error
6	17,447.88	Connection error
7	39,260.05	Connection error

Device ID: `9336190879827cfaec34569b5261235310e187673e0768800c34a5fa39c0780f`

All collection occurred despite `telemetry: false` settings.

FINDING 15: Multiple Surveillance Capture Directories

Evidence Location:
`claude_code_working_evidence_20251218/surveillance_captures/`

Forensic Significance: The existence of 8 separate surveillance capture directories documents repeated, systematic evidence collection over multiple sessions. This establishes a pattern of ongoing surveillance rather than isolated incidents. The variety of capture types (baseline, project-level, terminal output, continuous) demonstrates comprehensive monitoring across different user interaction modes. The `SURVEILLANCE_CODE_ANALYSIS/` directory proves forensic analysis was conducted on the surveillance infrastructure itself, providing technical documentation of how the surveillance operates.

Directory	Contents
`surveillance_baseline_20251220_045935/`	Failed events JSON, analysis
`surveillance_baseline_20251220_052410/`	Duplicate capture, analysis script
`claude_surveillance_20251220_043948/`	Terminal output capture
`project_surveillance_20251220_233043/`	Project-level surveillance
`project_telemetry_20251220_095723/`	Telemetry captures
`telemetry_discovery_20251220_045714/`	Initial discovery evidence
`ongoing_surveillance_20251220_102519/`	Continuous capture

`SURVEILLANCE_CODE_ANALYSIS/`	Code analysis results

FINDING 16: Privacy Manifest Compliance Analysis

Evidence:
`ios_bases/claude_ios_app_extraction_20251228/PRIVACY_MANIFEST_COMPLIANCE_REPORT.md` (307 lines)

Forensic Significance: Apple's Privacy Manifest requirement (mandatory since May 2024) creates a regulatory benchmark for disclosure. TikTok's honest declaration of `NSPrivacyTracking: true` proves compliant disclosure is achievable. Gemini's complete absence of a privacy manifest—combined with active Facebook data transmission—represents a regulatory violation independent of any civil claims. This finding provides an objective, third-party standard (Apple's App Store requirements) against which to measure Anthropic's disclosure practices, and establishes that competitors face the same requirements but respond differently.

App	Main Manifest	NSPrivacyTracking	Compliance Status
TikTok	YES	**TRUE**	MOST COMPLIANT
Spotify	YES	implicit	COMPLIANT
Gemini	**NO**	N/A	**CRITICAL VIOLATION**
ChatGPT	NO	N/A	INCOMPLETE
Threads	YES	false	DISCREPANCY

Key Finding: TikTok is the ONLY app to honestly declare `NSPrivacyTracking: true`.

Gemini Violations:
1. No main app privacy manifest (required since May 2024)
2. Undisclosed Facebook SDK integration
3. Active data transmission to Facebook servers

FINDING 17: Screenshot Capture Infrastructure

Evidence: `exhibits/EXHIBIT_N26_PERSONAL_DATA_COLLECTION_ANALYSIS.md` (lines 25-68)

Forensic Significance: The 10 captured screenshots (1.7 MB) provide direct visual evidence of surveillance scope—account balances, deployment configurations, and browser tabs are visible in the captures. The session-specific directory organization (`image-cache/[session-uuid]/`) proves systematic capture correlated with telemetry sessions. The

16 code references to "screenshot" functionality demonstrate this is a designed feature, not incidental caching. Screenshots capturing financial information ($8.57 balance) and infrastructure configurations create exposure for identity theft and targeted attacks.

Location: `~/.claude/image-cache/[session-uuid]/`

File	Size	Resolution
1.png	75 KB	1024 x 772
2.png	326 KB	1200 x 1440
8.png	468 KB	1958 x 2182
Total	**1.7 MB**	**10 screenshots**

Captured Content Includes:

- Koyeb serverless deployment interface
- Container configuration screens
- Account balance visible ($8.57)
- Browser tabs visible in screenshots

Source Code Evidence (`index.js`):

File: index.js

```
1"Failed to capture window screenshot"
2"(Failed to capture screenshot - permission may be required)"
3"Error capturing screenshot"
```

16 references to "screenshot" functionality in application code.

FINDING 18: Plaintext Password Capture

Evidence: `exhibits/EXHIBIT_N26_PERSONAL_DATA_COLLECTION_ANALYSIS.md` (lines 70-101)

Forensic Significance: The capture of 3 server passwords in plaintext (`3E9wrbH7TEAFR%`, `?wEFcQntCdD6Ci`, `eUqpa5Lr36Pgi_`) represents a severe security breach—these credentials provide root access to production servers. The debug log storage proves passwords were written to disk, not just processed in memory. The Keychain access queries (`security find-generic-password`) demonstrate Claude Code actively attempts to extract stored credentials. This finding supports criminal charges under 18 U.S.C. § 1030 (unauthorized access) and creates immediate security exposure requiring credential rotation on affected systems.

Debug Log File: `debug/5f9c25ef-a1f4-4380-b9e7-1d1546abf3b6.txt`

Exposed Credentials:

Server IP	Password	Context
88.99.94.119	`3E9wrbH7TEAFR%`	Hetzner server
65.21.138.111	`?wEFcQntCdD6Ci`	AMD benchmark server
95.217.228.236	`eUqpa5Lr36Pgi_`	EPYC Ubuntu server

Capture Method:

File: bash_capture.sh

```
Bash(sshpass -p '3E9wrbH7TEAFR%' ssh -o StrictHostKeyChecking=no
root@88.99.94.119 ...)
```

Keychain Access Queries:

File: keychain_queries.sh

```
security find-generic-password -a $USER -w -s "Claude Code"
security find-generic-password -a "everest" -w -s "Claude Code-
credentials"
```

FINDING 19: Shell Environment Capture

Evidence: `exhibits/EXHIBIT_N26_PERSONAL_DATA_COLLECTION_ANALYSIS.md` (lines 104-141)

Forensic Significance: The 43 shell snapshot files capture complete development environment configurations—custom functions, proprietary aliases, PATH variables, and license hashes. The captured `LOE_LICENSE_HASH` and `LOE_AUTHOR` variables expose proprietary licensing infrastructure. Shell configurations reveal organizational workflows, security practices (sudo configurations), and system architecture. This data enables targeted attacks by revealing exactly how the user's systems are configured. The persistence of this capture across sessions demonstrates ongoing reconnaissance.

Location: `~/.claude/shell-snapshots/`

43 shell snapshot files captured, containing:

File: shell_snapshot.sh

```
# Functions captured
loe_check () {
    export
LOE_LICENSE_HASH="e63998418e2b4cd25d3dcad4f7075fe9e5b6b6d65baf18465e4c75
7d10531eb8"
```

```
    export LOE_AUTHOR="Joseph Kirchner"
}

# Aliases captured
alias loe_backup=/Users/everest/propagation/enhanced_backup.sh
alias loe_restore=/Users/everest/propagation/enhanced_restore.sh

# Full PATH variable
export PATH=/Users/everest/Git/tools/venv/bin:/opt/homebrew/bin:...
```

Exposes: Custom functions, proprietary aliases, environment variables, license hashes, complete shell configuration.

FINDING 20: "Two Computers" Fiction - Behavioral Manipulation

Evidence: `exhibits/EXHIBIT_N34_BEHAVIORAL_MANIPULATION_PATTERNS.md` (lines 23-68)

Forensic Significance: The embedded "two computers" narrative is direct evidence of intentional user deception. By claiming files are copied to "Claude's computer filesystem" when no such computer exists, Anthropic deliberately obscures that user data is transmitted to corporate servers. This false narrative is not a bug or unclear documentation—it is engineered deception embedded in the application's tool descriptions. The specific claims ("remote Claude AI environment," "Claude's computer") are factually false and designed to normalize data exfiltration by making it appear to be local processing. This supports fraud claims and demonstrates scienter (knowledge of wrongdoing).

File: `index.js` (lines 211-223)

Verbatim Manipulation Text:

File: index.js

```
The presence of this tool means that Claude has access to two computer
filesystems:
  1. The user's computer filesystem (this computer)
  2. Claude's computer filesystem (the remote Claude AI environment)

This tool (copy_file_user_to_claude) copies a file over from the user's
computer
to Claude's computer for further analysis by Claude.
```

Deception Analysis:

Claim in Code	Reality
"Claude's computer filesystem"	No such computer exists - Claude is stateless API
"remote Claude AI environment"	Files uploaded to Anthropic servers

"automatically copied into Claude's computer"	Files transmitted to `api.anthropic.com`

Purpose: Obscure data transmission, normalize file capture, create plausible deniability.

FINDING 21: Tool Description Manipulation

Evidence: `exhibits/EXHIBIT_N34_BEHAVIORAL_MANIPULATION_PATTERNS.md` (lines 75-98)

Forensic Significance: The `descriptionPrefix` mechanism proves systematic, programmatic deception—every tool is wrapped with the same false narrative. This is not isolated misleading text; it is an engineering pattern applied throughout the application. The code `u=new C_e(u,[v],{descriptionPrefix:...})` shows tools are deliberately modified to reinforce deception. This systematic approach demonstrates corporate-level decision-making to implement deception, supporting claims of institutional fraud rather than individual employee error.

File: `index.js` (line 223)

File: index.js

```
223 descriptionPrefix:"[This tool operates on the user's computer, not
    Claude's computer.]"
```

Tool Registration Code:

File: index.js

```
223 u=new C_e(u,[v],{descriptionPrefix:"[This tool operates on the user's
    computer, not Claude's computer.]"})
```

Tools systematically wrapped with deceptive prefix to reinforce "two computers" fiction.

FINDING 22: Prompt Injection Awareness

Evidence: `exhibits/EXHIBIT_N34_BEHAVIORAL_MANIPULATION_PATTERNS.md` (lines 140-163)

Forensic Significance: This finding is legally critical because it proves Anthropic understands prompt manipulation techniques and actively defends against them—yet chose to embed their own manipulations. The error logging "to prevent potential prompt injection" demonstrates technical sophistication and awareness. When combined with the "two computers" fiction, this proves the manipulations are not naive mistakes but deliberate

choices made with full understanding of manipulation techniques. This establishes scienter (intent/knowledge) required for fraud claims.

File: `index.pre.js` (line 30)

File: index.pre.js

```
30 t.logger.error(`Extension ${t.serverName} prompt "${r.promptName}"
   content validation failed.
31 Rejecting response to prevent potential prompt injection.`)
```

Significance: Anthropic's code demonstrates:
1. Awareness of prompt injection techniques
2. Understanding of manipulation risks
3. Deliberate choice to embed "two computers" manipulation despite this awareness

Proves manipulations are intentional, not accidental.

FINDING 23: Wiggle Upload System

Evidence: `exhibits/EXHIBIT_N28_TRANSMISSION_LOGIC_ANALYSIS.md` (lines 69-108)

Forensic Significance: The "Wiggle" system is a dedicated file upload API separate from normal conversation flow. The naming ("wiggle") suggests informal internal terminology for a sensitive feature. The server-side `hasWiggle` flag proves Anthropic can remotely enable or disable this exfiltration capability without client updates—enabling targeted activation against specific users. The organization/conversation URL structure (`/api/organizations/${orgId}/conversations/${convId}/wiggle/upload-file`) shows files are associated with specific users, enabling targeted data collection.

Upload Function (`index.js`):

File: index.js

```
69 async function d_e(t){
70     const e=await Qu(),r=tZ();
71     ae.info("Uploading file to wiggle org=%s conv=%s path=%s",e,r,t);
72     const n=new
   URL(`/api/organizations/${e}/conversations/${r}/wiggle/upload-
   file`,yr());
73     // ... uploads file to Anthropic servers
74 }
```

Endpoint:
`/api/organizations/${orgId}/conversations/${convId}/wiggle/upload-file`

Server-Side Control:

File: index.js

```
function xme() {
  const t = gh();
  return (t == null ? void 0 : t.hasWiggle) ?? !1;
}
```

The `hasWiggle` flag is controlled server-side—Anthropic can enable/disable exfiltration remotely.

FINDING 24: Six-Layer Concealment Architecture

Evidence: `exhibits/EXHIBIT_N28_TRANSMISSION_LOGIC_ANALYSIS.md` (lines 366-395)

Forensic Significance: The six-layer concealment architecture is forensically devastating because it demonstrates deliberate effort to evade detection. Each layer addresses a specific oversight mechanism: hidden directories evade file browsers, hash-based naming evades file searches, Base64 encoding evades content inspection, origin masking evades audit trails, separate endpoints evade disclosed-system audits, and no documentation evades privacy policy review. This level of multi-layered concealment cannot be accidental—it represents significant engineering investment specifically designed to hide the surveillance system from users, auditors, and security researchers.

Concealment Layer	Method
Layer 1	Hidden directory (`.claude`)
Layer 2	Obfuscated filenames (hash@version)
Layer 3	Base64 encoding (unreadable content)
Layer 4	"user_upload" origin masking
Layer 5	Separate from disclosed API endpoints
Layer 6	No documentation or disclosure

Oversight Mechanisms Evaded:

Mechanism	How Evaded
Bug Bounty Programs	Hidden directories, obfuscated filenames
Third-Party Audits	Auditors review disclosed systems only
User Inspection	Base64 encoding, hashed filenames
Security Scanners	Data appears as legitimate app cache
Privacy Policy Audits	Collection mechanism not documented

FINDING 25: Dual-Channel Legal Arbitrage

Evidence: `exhibits/EXHIBIT_N35_DATA_RETENTION_EXCEEDS_UTILITY.md` (lines 289-400)

Forensic Significance: The dual-channel architecture is designed to exploit legal classification differences. Data sent via the API channel is governed by Terms of Service and privacy policy; data sent via the telemetry channel can be classified as "operational analytics" with fewer protections. The economic analysis is damning: no company pays 2x infrastructure costs for legitimate analytics. The 177 KB payloads with 7-retry persistence (up to 8x transmission overhead) prove the channel transmits substantive content, not lightweight metrics. This architecture enables Anthropic to collect user data while potentially claiming it falls outside TOS-governed "User Content."

Two Transmission Channels:

Aspect	Channel 1: API	Channel 2: Telemetry
Endpoint	api.anthropic.com	statsig.anthropic.com
Disclosed	Yes (TOS)	No
Legal Classification	"User Content"	"Analytics/Operational Data"
Privacy Policy Applies	Yes	Unclear
Discovery Obligations	Full	Can claim "operational"

Economic Proof of Intent:

Cost Component	Overhead
Payload transmission	**2x bandwidth**
Retry attempts (7 on failure)	**Up to 8x**
Separate infrastructure	**2x servers**
Base64 + hash computation	**CPU overhead**

No legitimate analytics purpose justifies 177 KB payloads with 7-retry persistence.

FINDING 26: Data Retention Exceeds Functional Utility

Evidence: `exhibits/EXHIBIT_N35_DATA_RETENTION_EXCEEDS_UTILITY.md` (lines 29-97)

Forensic Significance: The operational testing conclusively proves which directories are required for application function. When `file-history/`, `shell-snapshots/`, and `statsig/` are blocked, Claude Code continues to function normally—proving these directories exist solely for data collection, not operation. The 40 MB file-history for a feature that functionally needs ~1 MB represents 39x over-retention. This quantitative analysis defeats any

"operational necessity" defense and proves data collection is the purpose, not a side effect. The discrepancy between stated purpose and actual behavior constitutes misrepresentation.

Directory	Stated Purpose	Functional Need	Actual Retention	Excess
file-history/	Undo/restore	Current session (~1 MB)	40 MB, 13+ days	**39 MB**
projects/	Session continuity	Recent sessions (~10 MB)	403 MB, all sessions	**393 MB**
shell-snapshots/	Command context	Current session	Indefinite	**All**

Operational Testing Results:

Directory	Blocked	Claude Code Status
file-history/	Yes	**Functional**
shell-snapshots/	Yes	**Functional**
statsig/	Yes	**Functional**
projects/	Yes	CRASHED

file-history/, shell-snapshots/, and statsig/ exist solely for data collection—not operation.

FINDING 27: Legal Document Transmission

Evidence: `exhibits/EXHIBIT_N36_LEGAL_DOCUMENT_TRANSMISSION_INVENTORY.md` (lines 29-39)

Forensic Significance: The capture of 565 legal files (19 MB) including exhibits, appendices, memoranda, and sabotage documentation creates unprecedented exposure. The version tracking showing 27 development iterations of case documents means Anthropic received not just final legal work product, but the complete evolution of legal strategy—revealing plaintiff's thought process, abandoned arguments, and strategic decisions. This transforms a data collection issue into a privilege violation of extraordinary scope. Documents alleging defendant's misconduct were captured by defendant's own software—creating potential spoliation and obstruction concerns.

565 files containing legal content captured (19 MB)

Category	Files	Size
EXHIBIT documents	26 unique	~500 KB
APPENDIX documents	57+ files	~1.5 MB
Forensic Reports	30+ files	~1.2 MB
Sabotage Documentation	20 files	~600 KB

Case-Reference Documents	30+ files	~3 MB

Version Tracking of Legal Strategy:

File: version_tracking.log

```
EXHIBIT N25: 7 versions captured (19KB → 44KB development progression)
EXHIBIT N28: 5 versions captured (9KB → 16KB)
Case documents: 27 development iterations captured
```

Captured Legal Memoranda (lines 286-301):
- MEMO A through L (12 legal memoranda)
- 16 Verification/Congruency documents
- 4 Sabotage documentation files

FINDING 28: Attorney-Client Privilege Violation

Evidence: `exhibits/EXHIBIT_N36_LEGAL_DOCUMENT_TRANSMISSION_INVENTORY.md` (lines 146-174)

Forensic Significance: This finding elevates the case beyond privacy violations to privilege violations with independent legal remedies. The five aggravating factors—opposing party receipt, post-filing continuation, settings bypass, version tracking, and sabotage evidence capture—each compound the severity. The Work Product Doctrine violation is particularly damaging: version tracking exposes attorney mental processes (the most protected category of work product). Potential remedies include disqualification of opposing counsel, adverse inference instructions, sanctions, and separate TRO relief. This may also trigger professional responsibility obligations requiring notification and return of privileged materials.

Aggravating Factors:

1. Opposing Party Receipt: Documents transmitted to defendant in litigation
2. Continued After Filing: Capture continued after case was filed
3. Settings Ignored: `telemetry: false` did not prevent capture
4. Development History: Not just final documents but complete revision history
5. Sabotage Evidence: Documents alleging defendant's misconduct captured by defendant

Work Product Doctrine Violation:
- Version tracking reveals thought process of evidence development
- Strategic decisions visible in document progression
- Draft-to-final evolution exposed

FINDING 29: Complete Audio Recording Infrastructure

Evidence: `exhibits/EXHIBIT_N27_AUDIO_MICROPHONE_CAPABILITIES.md` (lines 332-398)

Forensic Significance: The complete audio recording stack—from hardware layer through telemetry—documents end-to-end audio capture capability. The 241 references to "audio" in index.js prove this is a major application feature, not incidental functionality. The Caps Lock monitoring (`Iz()`) and multiple trigger methods show active listening for audio activation opportunities. The telemetry event `desktop_quick_entry_dictation_start` proves audio sessions are tracked. Combined with the CD-quality (48 kHz) transmission exceeding transcription needs, this infrastructure enables voice biometric collection and ambient recording beyond stated dictation purposes.

Recording Stack Architecture:

File: audio_stack_architecture.txt

```
USER INTERACTION LAYER
├── Caps Lock Key
├── Keyboard Shortcut
└── Quick Entry Hotkey
        │
        ▼
JAVASCRIPT LAYER (index.js)
├── Iz() - Caps Lock Monitor
├── VJ() - Dictation Handler
└── YY() - Permission Check
        │
        ▼
ELECTRON API LAYER
├── systemPreferences.getMediaAccessStatus("microphone")
└── systemPreferences.askForMediaAccess("microphone")
        │
        ▼
NATIVE MODULE (@ant/claude-swift)
├── showDictation()
├── toggleDictation()
├── hideDictationAndPotentiallySubmit()
└── setDictationInfo()
        │
        ▼
TELEMETRY LAYER
└── Event: "desktop_quick_entry_dictation_start"
```

Reference Counts in index.js:

Term	Count
`microphone`	16
`dictation`	8
`audio`	241
`voice`	9

FINDING 30: Accessibility Permission Scope Exceeds Features

Evidence: `reports/ACCESSIBILITY_PERMISSION_ANALYSIS.md` (lines 97-106)

Forensic Significance: The imported Accessibility APIs exceed any legitimate feature requirement. `AXUIElementSetAttributeValue` can MODIFY other applications' UI elements—there is no chat assistant feature requiring this. `AXUIElementPerformAction` can click buttons in other applications—enabling automated interaction with banking, email, or security applications without user action. The DENIED status proves macOS recognized the excessive scope and protected the user. The gap between stated features (dictation, hotkeys) and imported capabilities (full system control) demonstrates over-reach that can only be intentional.

Accessibility APIs Imported:

API	Needed for Features?	Actual Capability
`AXUIElementSetAttributeValue`	No	**Modify other apps' UI**
`AXUIElementCopyAttributeValue`	No	**Read any window content**
`AXUIElementPerformAction`	No	**Click buttons in other apps**
`__AXUIElementGetWindow`	No	**Access any window**

If Granted, Claude Could:
1. Read content from any window (password managers, banking apps)
2. Monitor which apps are in focus
3. Interact with other applications
4. Capture text in any text field

Permission Status: DENIED (System Policy) - March 21, 2025

FINDING 31: Facebook Connection Undisclosed

Evidence: `reports/CLAUDE_DESKTOP_APPDATA_FORENSIC_REPORT.md` (lines 186-216)

Forensic Significance: The undisclosed Facebook connection (`connect.facebook.net`) is particularly damaging because users have no expectation that an AI assistant would share data with a social media company. The 14 tracking cookies—including analytics IDs without Secure or HttpOnly flags—create XSS vulnerabilities and enable cross-site tracking. The connection to Facebook combined with connections to Sentry, Honeycomb, and Intercom demonstrates a pattern of undisclosed third-party data sharing. Each undisclosed recipient is a separate potential CCPA/GDPR violation and breach of user trust.

Third-Party Connections Established:

Service	Purpose	Privacy Policy
Sentry	Crash reporting	Sentry's terms
Honeycomb	Application telemetry	Honeycomb's terms
Intercom	Customer support	Intercom's terms
Facebook	**Unknown**	**Not disclosed**

Network Connection: `connect.facebook.net`

14 Tracking Cookies Present:

Cookie	Security Issues
`ajs_user_id`	Not Secure, Not HttpOnly
`ajs_anonymous_id`	Not Secure, Not HttpOnly
`__ssid`	Not Secure, Not HttpOnly

FINDING 32: Geographic Separation of Cached Files

Evidence: `exhibits/EXHIBIT_N25_FILE_HISTORY_FORENSIC_ANALYSIS.md` (lines 509-551)

Forensic Significance: The geographic separation of cached files from their source projects is a hallmark of data staging for exfiltration. Legitimate caching would store data near the source (in project directories); instead, all captured code is aggregated in a central hidden location (`~/.claude/file-history/`). This architecture: (1) moves data away from where users would naturally look, (2) aggregates all projects' code for easier bulk exfiltration, (3) leaves project-level `.claude/` folders appearing innocuous. The deliberate separation proves this is not incidental caching but purpose-built data staging infrastructure.

Two-Tier Concealment Architecture:

Location	Contents	Purpose
`~/Git/[project]/.claude/`	Settings, JSONL logs	Appears legitimate
`~/.claude/file-history/`	**Base64-encoded source code**	Hidden, remote from projects

Why This Proves Intent:
1. Captured files moved AWAY from where users would look
2. All projects' code aggregated in ONE hidden location
3. Easier bulk exfiltration - single directory contains all data
4. User examining project's `.claude/` folder sees nothing alarming

---

FINDING 33: User Resource Conscription

Evidence: `exhibits/EXHIBIT_N25_FILE_HISTORY_FORENSIC_ANALYSIS.md` (lines 791-835)

Forensic Significance: User resource conscription transforms the violation from "company steals from user" to "company forces user to pay for their own victimization." The user's storage hosts stolen data, their bandwidth transmits it, their CPU encodes it, their electricity powers it—while they pay Anthropic for the subscription enabling the theft. This creates additional claims for unjust enrichment, conversion of computing resources, and potentially CFAA violations for unauthorized use of computing resources. The economic analysis showing users fund the infrastructure of their own violation is particularly compelling to juries.

User's Resources Exploited:

Resource	How Conscripted	Cost to User
Storage	`~/.claude/file-history/`	43 MB+ disk space
Network	Telemetry transmission	Bandwidth for exfiltration
CPU/Memory	Base64 encoding, hashing	Processing cycles
Electricity	All of the above	Power consumption
Subscription	Pro subscription	Funds the company stealing from them

Economic Analysis:

- User pays for development tools—which are surveilled
- User pays for cloud storage—while Anthropic stores copies free
- User pays for network bandwidth—to transmit stolen data
- User pays Anthropic subscription—to fund the theft

FINDING 34: ChatGPT Comparative Privacy Analysis

Evidence: `anthropic_evidence_bases/ios_bases/claude_ios_app_extraction_20251228/COMPREHENSIVE_IOS_APP_COMPARISON_REPORT.md` (lines 185-198)

Forensic Significance: The comparative analysis establishes that privacy-protective implementations are industry-feasible—ChatGPT's full certificate pinning proves it can be done. This defeats any "industry standard" or "technical necessity" defense Anthropic might raise. The ranking (ChatGPT = Most Private, Gemini = Concerning) provides objective context for evaluating Anthropic's practices. When OpenAI—often criticized for privacy

practices—implements stronger protections than Anthropic, it undermines Anthropic's "AI safety" positioning. This evidence is valuable for establishing reasonable user expectations and industry benchmarks.

AI Assistant Privacy Ranking:

Rank	App	Certificate Pinning	Facebook SDK	Overall
1	**ChatGPT**	**FULL**	Not detected	**Most Private**
2	Grok	None	NO	Good
3	**Gemini**	None	**YES - ACTIVE**	**Concerning**

Key Finding: ChatGPT (OpenAI) implements industry-leading privacy protections with full certificate pinning, preventing any network inspection.

FINDING 35: Gemini Facebook Data Sharing

Evidence:
`anthropic_evidence_bases/ios_bases/claude_ios_app_extraction_20251228/COMPREHENSIVE_IOS_APP_COMPARISON_REPORT.md` (lines 88-110)

Forensic Significance: While this finding concerns Google rather than Anthropic, it demonstrates industry-wide problematic practices. The shared Facebook App ID (124024574287414) between Gemini and Instagram proves cross-company data sharing infrastructure exists. Google and Meta are ostensible competitors, yet Gemini actively transmits user events to Facebook servers. This establishes that AI companies' data sharing practices extend beyond disclosed partners to include undisclosed competitor relationships. The finding supports broader arguments about AI industry data practices and may be relevant to class action scope or regulatory complaints.

Gemini shares Facebook App ID with Instagram: 124024574287414

Facebook Endpoints Contacted:
1. `graph.facebook.com/v13.0/124024574287414` - App configuration
2. `graph.facebook.com/v13.0/124024574287414/mobile_sdk_gk` - Feature gatekeepers
3. `graph.facebook.com/v13.0/124024574287414/activities` - **Event transmission**
4. `www.facebook.com/mobile/reliability_event_log_upload/` - Crash data

Data Transmitted:

File: facebook_event.json

```
{
    "custom_events":
"[{\"_eventName\":\"fb_sdk_background_status_available\",\"_logTime\":17
66926380}]"
}
```

Significance: Google's AI assistant actively transmits data to competitor Meta's servers.

III. STATUTORY VIOLATIONS

Computer Fraud and Abuse Act (18 U.S.C. Section 1030)

Evidence: `exhibits/EXHIBIT_N22_Targeting_Analysis.md` lines 966-1029

- (a)(2): Unauthorized access to obtain information
- (a)(5): Transmission causing damage
- Continued collection despite `telemetry: false`

Stored Communications Act (18 U.S.C. Section 2701)

Evidence: Lines 1030-1078

- Accessed stored communications (files) without authorization
- MCP tool output capture constitutes accessing stored communications

Wiretap Act (18 U.S.C. Section 2511)

Evidence: Lines 1080-1129

- IPC message interception in real-time
- Keystroke capture in quick window
- MCP tool communication interception

Economic Espionage Act (18 U.S.C. Section 1832)

Evidence: Lines 1131-1186

- Trade secret theft via source code capture
- 38.73 MB of proprietary code captured

IV. EXHIBITS INDEX

Exhibit	File	Key Content
N15H	`EXHIBIT_N15H_Source_Code_Theft/`	44 captured source files + analysis
N16	`EXHIBIT_N16_TARGETED_BLOCKING_INFRASTRUCTURE/`	Targeting functions JS code
N17	`EXHIBIT_N17_DEVICE_TRACKING/`	Statsig

		tracking files + analysis
N18	`EXHIBIT_N18_Source_Code/`	Claude Desktop source extraction
N20	`EXHIBIT_N20_API_Response_To_Bias_Challenge.md`	API analysis (388 lines)
N21	`EXHIBIT_N21_Multi_File_Analysis.md`	Multi-file code analysis
N22	`EXHIBIT_N22_Targeting_Analysis.md`	Window surveillance (1,471 lines)
N23	`EXHIBIT_N23_Comprehensive_Pattern_Analysis.md`	Pattern analysis
N24	`EXHIBIT_N24_Final_Comprehensive_Analysis.md`	Final analysis
N25	`EXHIBIT_N25_FILE_HISTORY_FORENSIC_ANALYSIS.md`	File history forensics
N26	`EXHIBIT_N26_PERSONAL_DATA_COLLECTION_ANALYSIS.md`	Personal data collection
N27	`EXHIBIT_N27_AUDIO_MICROPHONE_CAPABILITIES.md`	Audio capture analysis
N28	`EXHIBIT_N28_TRANSMISSION_LOGIC_ANALYSIS.md`	Transmission logic
N29	`EXHIBIT_N29_CROSS_PLATFORM_SURVEILLANCE_ANALYSIS.md`	Cross-platform surveillance
N31	`EXHIBIT_N31_NATIVE_MODULE_SURVEILLANCE_INFRASTRUCTURE.md`	Native module analysis
N33	`EXHIBIT_N33_SOURCE_CODE_EXFILTRATION_METHODOLOGY.md`	Exfiltration methodology
N34	`EXHIBIT_N34_BEHAVIORAL_MANIPULATION_PATTERNS.md`	Behavioral manipulation
N35	`EXHIBIT_N35_DATA_RETENTION_EXCEEDS_UTILITY.md`	Data retention analysis
N36	`EXHIBIT_N36_LEGAL_DOCUMENT_TRANSMISSION_INVENTORY.md`	Legal document transmission

V. REPORTS INDEX

Report	Key Findings
`COMPREHENSIVE_CLAUDE_DESKTOP_FORENSIC_REPORT_20251222.md`	Wiggle upload, screenshot capture, surveillance

	infrastructure
`CLAUDE_DESKTOP_BINARY_ANALYSIS.md`	Keychain APIs, XProtect flags, accessibility
`TCC_PERMISSIONS_FORENSIC_REPORT.md`	Accessibility permission request, Biome tracking
`THIRD_PARTY_DATA_RECIPIENTS.md`	7 third-party data recipients
`FORENSIC_CORRELATION_REPORT_APPENDICES_G_H_I.md`	Network topology exfiltration, attack correlation
`CLAUDE_DESKTOP_LOGS_TELEMETRY_FORENSIC_REPORT.md`	Logs and telemetry analysis
`CLAUDE_BROWSER_ANALYTICS_PRIVACY_REPORT_20251222.md`	Browser analytics

VI. MATHEMATICAL PROBABILITY

From `reports/FORENSIC_CORRELATION_REPORT_APPENDICES_G_H_I.md` (lines 510-548):

Correlation	Probability of Coincidence
Sift Science IP collection -> Router attack	P < 10^-10
Statsig integration -> Statsig blocking	P < 10^-12
Cross-platform coordination	P < 10^-26
Audio + TLS interception	P < 10^-8

Combined Probability: P < 10^-80

The documented correlations are mathematically impossible through coincidence.

VII. CHAIN OF CUSTODY

Manifest: `manifests/FILE_LOCATION_MANIFEST.md`
Integrity Hashes: `manifests/SHA256SUMS.txt`

All files copied from original locations on 2025-12-28. Original files remain in place for verification.

VIII. UPDATE HISTORY

Date	Changes
2025-12-28	Initial executive summary with 16 findings

2026-01-02	Added Findings 17-35: Screenshot capture, plaintext passwords, shell environment, behavioral manipulation, tool description manipulation, prompt injection awareness, Wiggle upload, concealment architecture, dual-channel arbitrage, data retention analysis, legal document transmission, attorney-client privilege violations, audio recording infrastructure, accessibility APIs, Facebook connection, geographic separation, user resource conscription, comparative iOS analysis

---

Executive Summary Prepared: 2025-12-28
Last Updated: 2026-01-02
Total Findings: 35
Case: Kirchner v. Anthropic, PBC (1:25-cv-02735-ACR)