AI Prompt/Input Activity Evidence Collection in Windows and Macintosh Systems

A practitioner-focused white paper for digital forensic investigators, incident responders, eDiscovery professionals, and DFIR tool developers.

Author: Christopher L. T. Brown (clbrown@techpathways.com), Technology Consulting in Digital Forensics, InfoSec & AI, September 2026

Executive Summary

Investigators increasingly need to determine whether a user on a Windows or macOS system submitted sensitive text, pasted content, uploaded files, or otherwise interacted with an AI application or AI web service, and, if so, what was submitted, when, and through which channel.[1][11][14][1][11][14] No single artifact answers this question completely. Evidence is distributed across desktop application storage, browser storage engines (IndexedDB, LevelDB, LocalStorage), OS-level execution and file-activity traces, clipboard/pasteboard remnants, and cloud-side account or audit logs the investigator may not control directly.[1][11][14][1][11][14]

This paper organizes that evidence into practitioner categories for Windows and macOS separately, ranks each artifact by evidentiary value (probative vs. corroborative) and volatility, and proposes a concrete forensic collection and reporting tool design ("PromptTrace," illustrative) detailed enough to guide a developer's proof of concept.[5][9][5][9] It closes with limitations that must accompany any finding in this domain: encryption and sandboxing, ephemeral data loss, the frequent absence of exact submitted text, and the difficulty of distinguishing AI-specific activity from ordinary browser/application use.[9][25][9][25]

Key finding: the single highest-value artifact category on both platforms is browser and desktop-app local storage (IndexedDB/LevelDB), because it is the only artifact family that can preserve actual prompt and response text rather than metadata about activity.[10][14][22][10][14][22] Everything else in this paper, clipboard, OS execution traces, provenance metadata, is best understood as corroboration for a timeline built around that core evidence, not a substitute for it.[16][32][16][32] For agentic and MCP-based tool use specifically, memory forensics is emerging as the only known recovery path when the agent or its host process leaves no disk log, an important exception to the general rule that disk-based artifacts are the primary evidence source.[48][48]

Scope and Investigative Problem Statement

Investigative question: did a user on this system submit prompts, paste sensitive text, upload files to, or download outputs from an AI application or AI web service, and can that activity be reconstructed with defensible evidence?

In scope: desktop AI applications (ChatGPT desktop, Claude desktop, Microsoft Copilot app, local LLM runners such as Ollama, LM Studio, and llama.cpp), browser-based AI web interfaces (ChatGPT, Claude.ai, Gemini, Copilot web), and the OS-level, browser-level, and cloud-sync traces those interactions may leave on Windows and macOS endpoints.[30][31][39][40][30][31][39][40]

Out of scope: mobile-device AI artifacts, network-layer packet capture/interception, server-side/cloud-provider internal logs (addressed only as a collection consideration requiring legal process), and content-authenticity/detection of AI-generated text or images (a related but distinct forensic problem).

Cross-reference: platform findings are in Windows Findings and macOS Findings; the comparative treatment of the same investigative question across platforms is in Comparative Analysis; the proposed tooling is in Proposed Tool Design.

Methodology and Evidence Standard

Findings in this paper are drawn from vendor documentation, peer-reviewed and preprint forensic research, and established DFIR practitioner references, cited inline.[1][11][14][39][40][1][11][14][39][40] Each claim is marked using one of three confidence tiers, consistent with the requirement to distinguish observed evidence from inference:

- **Confirmed / documented**, reported in vendor documentation, a peer-reviewed or preprint forensic study with a described methodology, or a well-established forensic reference, and not contradicted by other sources.[11][39][11][39]
- **Likely but unvalidated**, technically plausible and consistent with how the underlying platform or application framework is known to behave (for example, Electron apps generally using LevelDB-backed storage), but not independently confirmed for a specific vendor/version at the time of writing. Marked **[Likely]** in text and tables.
- **Inference requiring caution**, a conclusion an investigator might be tempted to draw from an artifact, flagged explicitly so it is not mistaken for direct evidence (for example, treating a browser history entry as proof of prompt content). Marked **[Inference]** in text.

Assumptions: the investigator has appropriate legal authority to examine the system; artifacts are being reconstructed from a forensic image, a live-response collection, or a combination of both; and AI vendor client applications and storage schemas change frequently enough that any specific file path or format should be treated as version-dependent and re-verified at exam time.[39][39][39][39]

Windows Findings

1. Installed AI Applications and Package/Install Evidence

- **What it may show:** that a desktop AI application (ChatGPT, Claude, Copilot, Ollama, LM Studio) was installed, when, and by which user profile.
- **Typical location(s):** `HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\` (per-user installs, which is how most consumer AI desktop apps install, critical because HKLM-only uninstall checks will miss them); Windows Event Log MSI/Application install events; `%ProgramFiles%` or `%LocalAppData%\\Programs\\<vendor>`.
- **Expected data format:** registry values (strings/binary); Windows Event Log (.evtx, structured XML).
- **Persistence/volatility:** Low volatility, persists until uninstall.

- **Collection considerations:** Always check per-user (`HKCU`) uninstall keys in addition to machine-wide (`HKLM`); a per-user-only install is easy to miss with default enterprise-inventory tooling.[32][32]
- **Interpretation caution:** proves installation and a install-time window, not usage or content.
- **Evidentiary value:** Corroborative, establishes capability and timeline anchor, not content.[5][32][5][32]

2. Application Logs, Configuration Files, and Local Databases

- **What it may show:** conversation history, cached prompts/responses, session tokens, application configuration.
- **Typical location(s):** ChatGPT and Claude desktop (Electron-based) write LevelDB-backed IndexedDB stores, e.g. `%UserProfile%\\AppData\\Local\\Packages\\OpenAI.ChatGPT-Desktop_<id>\\LocalCache\\Roaming\\ChatGPT\\IndexedDB\\https_chatgpt.com_0.indexeddb.leveldb\\`, documented directly, including the write-ahead-log (`.log`) mechanism that stages conversation writes before compaction.[14][14] Local LLM runners: Ollama writes `%LocalAppData%\\Ollama\\server.log`, `%UserProfile%\\.ollama\\models\\blobs\\`, and `%UserProfile%\\.ollama\\history` (CLI prompt history); LM Studio writes `%UserProfile%\\.lmstudio\\conversations\\{ID}.json` (full conversation JSON, described as containing "complete, timestamped user/AI conversation logs"), `%UserProfile%\\.lmstudio\\user-files\\{filename}` for uploaded files, and `%AppData%\\LM Studio\\logs\\main.log`.[39][40][39][40] Microsoft Copilot's Windows client is largely cloud-backed; local cache contents are less consistently documented and should be treated as **[Likely]** rather than confirmed until verified against the installed build.[33][34][33][34] A dedicated empirical study found that Windows 11 Copilot text queries persist in physical RAM in UTF-16LE format but are effectively sanitized from the virtual disk and pagefile, meaning memory acquisition, not disk imaging, is the reliable path to recovering Copilot prompt content.[33][33]
- **Expected data format:** LevelDB (Electron apps: `.log`, `.ldb`, `.sst` files), JSON (LM Studio), plain text (Ollama server log); Copilot query fragments in RAM as UTF-16LE text.[14][33][39][40][14][33][39][40]
- **Persistence/volatility:** Medium. LevelDB compaction can supersede or drop stale `.log` entries; JSON conversation files persist until deleted by the app or user.[14][39][14][39]
- **Collection considerations:** LevelDB requires purpose-built parsing (see Proposed Tool Design), a raw text scan of the `.log` file can recover fragments but will miss compacted `.ldb`/`.sst` content and can misorder entries. Because AI desktop apps update frequently, confirm the installed version's schema at exam time rather than assuming path stability.[39][39]
- **Interpretation caution:** a `.log` file with no visible conversation content does not prove no conversation occurred, content may already be compacted into `.ldb`/`.sst` segments requiring full LevelDB reconstruction.[14][14]
- **Evidentiary value:** Highest, this category is the most direct source of actual prompt and response text on Windows.

3. Prompt/Input Traces and Text-Entry Remnants

- **What it may show:** text a user typed or pasted into an AI application's input field, sometimes recoverable even if the session was not saved.
- **Typical location(s):** unsent/draft input may be cached in the same LevelDB/IndexedDB stores as (2) depending on the app's autosave behavior; Windows Clipboard History database at `%AppData%\\Local\\ConnectedDevicesPlatform\\<UserProfile>\\ActivitiesCache.db` when Clipboard History and cross-device sync are enabled.[1][2][1][2]

- **Expected data format:** SQLite (ActivitiesCache.db); LevelDB (app draft state, where applicable).
- **Persistence/volatility:** High, Clipboard History requires the feature to have been explicitly enabled by the user (off by default); ordinary (non-history) clipboard content is volatile and typically recoverable only via live memory acquisition.[1][3][4][1][3][4]
- **Collection considerations:** verify Clipboard History was enabled before relying on `ActivitiesCache.db`; for live systems, prioritize memory acquisition to capture ordinary clipboard content before shutdown.[3][6][3][6]
- **Interpretation caution:** clipboard content proves something was copied/pasted, not necessarily that it was submitted into an AI system, corroborate with app-level evidence from (2).[4][4]
- **Evidentiary value:** Probative when present, but frequently unavailable (feature-dependent, high volatility).

4. File Upload and Download Evidence

- **What it may show:** that a file was uploaded to an AI service, or an AI-generated output (document, image) was downloaded.
- **Typical location(s):** the `Zone.Identifier` NTFS alternate data stream attached to downloaded files, recording the originating URL, security zone, and referrer where captured by the browser; browser download-history databases; partial-download extensions (`.crdownload`, `.part`) in the Downloads folder or `%TEMP%`.[37][38][37][38]
- **Expected data format:** `Zone.Identifier` is a structured text stream readable via `Get-Content -Stream Zone.Identifier` or forensic ADS-aware tools; browser download history is typically SQLite.
- **Persistence/volatility:** Low for `Zone.Identifier` (persists with the file until explicitly stripped or the file moves to a filesystem that drops ADS); higher for partial-download temp files.
- **Collection considerations:** ADS-aware acquisition and viewing tools are required, many default imaging/triage workflows do not surface alternate data streams unless explicitly configured to.[37][38][37][38]
- **Interpretation caution:** `Zone.Identifier` is set by the downloading application (typically the browser), not universally by every AI desktop app's own export function, absence does not rule out AI origin.[37][37]
- **Evidentiary value:** High when present, ties a specific file to a source URL and timestamp with strong evidentiary weight.[37][37]

5. Clipboard, Temporary File, Cache, and Autosave Artifacts

- **What it may show:** copy/paste activity involving AI input or output; transient autosave content from apps used alongside an AI workflow (e.g., a document draft pasted into a chat).
- **Typical location(s):** `%TEMP%`; Office autosave (`.asd`) files; browser cache folders within each browser profile.
- **Expected data format:** varies by source application; Office `.asd` is a proprietary binary format.
- **Persistence/volatility:** Medium-high, `%TEMP%` is routinely purged by the OS, applications, and disk-cleanup utilities.[29][29]
- **Collection considerations:** collect early in triage; temp-file survival windows are unpredictable and not governed by a fixed retention policy.[29][29]
- **Interpretation caution:** temp/autosave content shows document drafting activity, not necessarily that the content was submitted to an AI system, treat as corroborative context.
- **Evidentiary value:** Corroborative.

6. Browser Artifacts Related to AI Web Interfaces

- **What it may show:** actual prompt and response text (via IndexedDB/LocalStorage), account identity (via cookies/Login Data), generated output content (via cache), and session state.
- **Typical location(s):** within each browser's profile folder, Chrome/Edge: `IndexedDB\\` and `Local Storage\\leveldb\\` subfolders keyed by origin (e.g., `https_chatgpt.com_0.indexeddb.leveldb`); History, Cookies, and Login Data as separate SQLite databases in the same profile.[10][12][13][14][15][10][12][13][14][15]
- **Expected data format:** LevelDB (IndexedDB, Local Storage) for Chromium-based browsers; SQLite for history/cookies/login data.[13][14][13][14]
- **Persistence/volatility:** Medium, subject to storage-quota eviction, explicit "Clear Browsing Data," and low-disk-space auto-eviction; research on browser-based conversational-AI extensions and IndexedDB forensics for chat-style web apps (WhatsApp Web, Instagram) demonstrates that persistent structured conversation data is recoverable from IndexedDB when not yet evicted.[10][12][15][10][12][15]
- **Collection considerations:** highest-priority Windows artifact category given eviction risk; requires LevelDB-aware parsing per origin, since each AI vendor's front end uses its own internal schema that can change with front-end updates, schema-version detection should be part of any tooling (see Proposed Tool Design).[14][14]
- **Interpretation caution:** browser History alone shows that a URL was visited, it does not show what was typed or received; do not conflate history entries with prompt content. **[Inference]** risk: treating a visited-URL entry as proof of a specific prompt.
- **Evidentiary value:** Very high (IndexedDB/LocalStorage) for direct content; History/Cookies are corroborative for account identity and access timing only.

7. OS-Level Traces of Application Execution and Document Handling

- **What it may show:** that an AI-related application or file was executed/opened, how frequently, and through what shell path.
- **Typical location(s):** UserAssist (`NTUSER.DAT`, ROT13-encoded GUI execution history), ShellBags (folder-navigation history tied to the user's registry hive), Jump Lists and `.lnk` shortcut files (`%AppData%\\Microsoft\\Windows\\Recent\\AutomaticDestinations\\`), Prefetch (`C:\\Windows\\Prefetch\\`), AmCache (`C:\\Windows\\AppCompat\\Programs\\Amcache.hve`), ShimCache/AppCompatCache, and the Background Activity Moderator (BAM/DAM) registry keys, which record process execution including command-line invocations that UserAssist (GUI-launch-only) misses.[16][17][18][19][32][44][16][17][18][19][32][44]
- **Expected data format:** binary registry hives (UserAssist, ShellBags, AmCache); proprietary Jump List container format; Prefetch binary format.
- **Persistence/volatility:** Low-to-medium; these are among the more durable Windows execution artifacts, though Prefetch is disabled by default on some server SKUs and entries roll off with continued system use.
- **Collection considerations:** parse UserAssist/ShellBags/AmCache/ShimCache together to build an execution and file-access timeline; do not rely on any single one in isolation, since each has known interpretation limits (e.g., ShimCache does not reliably prove execution, only that a file was present and referenced).[17][18][17][18]

- **Interpretation caution:** these artifacts prove execution or folder access, not the content processed during that execution, they answer "was the AI application run and when," not "what was submitted." BAM records that a process ran, including via command line, but not its arguments or output.[16][19][44][16][19][44]
- **Evidentiary value:** Corroborative for timeline and attribution of activity to a user profile.

8. Persistence and Background Services

- **What it may show:** AI-related background services or agents configured to run automatically (relevant to agentic/automation use cases).
- **Typical location(s):** `Run`/`RunOnce` registry keys, Scheduled Tasks, Windows Services.[32][32]
- **Expected data format:** registry values; Task Scheduler XML; service configuration in the registry.
- **Persistence/volatility:** Low.
- **Collection considerations:** standard persistence-artifact triage applies; correlate with installed-application evidence (Category 1).
- **Interpretation caution:** presence of a persistence mechanism indicates the application is configured to run, not that any particular AI interaction occurred.
- **Evidentiary value:** Corroborative.

9. Network and Remote Interaction Traces

- **What it may show:** volume and timing of data sent to/received from an AI service's domain, even without content.
- **Typical location(s):** System Resource Usage Monitor (SRUM) database, `C:\\Windows\\System32\\sru\\SRUDB.dat`, which records per-application network bytes sent/received and timing.[5][32][5][32]
- **Expected data format:** ESE database (`.dat`).
- **Persistence/volatility:** Medium, SRUM retains a rolling window of data, not indefinite history.
- **Collection considerations:** correlate SRUM timing/volume spikes for a given application against browser/app-level content evidence to corroborate an upload or download event.
- **Interpretation caution:** SRUM proves volume and timing, never content, an upload spike to a browser process does not by itself identify what was uploaded or to which specific site if multiple tabs were open.
- **Evidentiary value:** Corroborative, valuable for timeline correlation and exfiltration-volume assessment.

10. Data Staging, Export, Sync, and Exfiltration Indicators; Search and Indexing Traces

- **What it may show:** deleted or exported document content that passed through Windows indexing before deletion; OneDrive sync state for files touched by an AI workflow.
- **Typical location(s):** Windows Search database `Windows.edb` (default path under `%ProgramData%\\Microsoft\\Search\\Data\\Applications\\Windows\\`), which can be carved for indexed records of deleted files, including partial content recovery in slack space; OneDrive local sync metadata.[35][36][35][36]
- **Expected data format:** Extensible Storage Engine (ESE) database.
- **Persistence/volatility:** Medium, indexed records of deleted files may persist in `Windows.edb` slack space after the source file is gone, but are not guaranteed to survive reindexing or long-term use.[35][35]
- **Collection considerations:** `Windows.edb` frequently requires ESE recovery/repair before parsing due to improper-shutdown states; budget for this in collection planning.[36][36]

- **Interpretation caution:** presence in the search index shows the file existed and was indexed, not that its content was submitted to an AI service, corroborate with (2) or (6).
- **Evidentiary value:** High specifically for recovering fragments of deleted document content otherwise unavailable.

11. Windows Recall and OS-Native AI Feature Snapshots

- **What it may show:** periodic screen snapshots and associated text/OCR index that could capture an AI chat session displayed on screen.
- **Typical location(s):** local snapshot database on Copilot+ PCs with Recall enabled (path and schema are Microsoft-controlled and subject to change; treat as **[Likely]**/version-dependent rather than a fixed reference path).[33][33]
- **Expected data format:** proprietary snapshot store with an associated searchable text/OCR index.
- **Persistence/volatility:** Medium, Recall is opt-in by default on supported hardware at time of writing and can be disabled or have history cleared by the user; retention window is configurable.
- **Collection considerations:** confirm hardware eligibility (Copilot+ PC) and whether Recall was enabled before assuming this artifact exists; do not assume presence on standard Windows 11 hardware.
- **Interpretation caution:** Recall captures what was visually on screen, not the underlying application data structure, it can corroborate that an AI chat interface was open and roughly what it displayed, but is not a substitute for parsing the application's own storage.
- **Evidentiary value:** Corroborative, visually direct when present, but availability is hardware- and configuration-dependent.

12. Evidence of Generated Output Handling

- **What it may show:** how a user handled AI-generated content after receiving it, saved, copied elsewhere, printed, or shared.
- **Typical location(s):** combination of File Upload/Download Evidence (Category 4), Clipboard artifacts (Category 3/5), and OS execution traces (Category 7) applied to the output file rather than the input.
- **Collection considerations:** treat generated-output handling as a correlation exercise across existing categories rather than a distinct artifact source.
- **Interpretation caution:** a saved or printed AI output proves handling of that specific file, not the full prompt history that produced it.
- **Evidentiary value:** Corroborative, dependent on correlation quality.

13. Anti-Forensic or Artifact-Reduction Behaviors

- **What it may show:** deliberate clearing of AI conversation history, browser data, or clipboard history; use of private/incognito browsing for AI web access; disabling of Clipboard History or Recall.
- **Typical location(s):** absence of expected artifacts combined with corroborating evidence of application use (e.g., installed-application evidence with no matching conversation data); browser "Clear Browsing Data" event traces where logged; private-browsing sessions generally leave minimal or no persistent local storage by design.
- **Interpretation caution:** absence of an artifact is not proof of anti-forensic intent, it may simply reflect normal storage eviction, a disabled feature, or a version that never wrote the artifact in the first place. Frame gaps as investigative questions, not conclusions.

- **Evidentiary value:** Inference-level; requires strong corroboration before being characterized as intentional evidence elimination.

14. Memory Forensics for Agentic and MCP (Model Context Protocol) Activity

- **What it may show:** tool-invocation traces for agentic AI systems and coding assistants that communicate over the Model Context Protocol (MCP), including tool discovery, invocation arguments, execution outcomes, and request/response pairs, identifiable via their JSON-RPC message structure, even when no disk-based log exists.
- **Typical location(s):** volatile process memory (RAM) of the host application (browser, IDE, or agent runtime); not disk-resident by default. A DFRWS USA 2026 study demonstrated reconstruction of MCP activity purely from memory acquisition when the associated application (tested against Composer 1 and Gemini 3 Flash) wrote no corresponding disk log.[48][48]
- **Expected data format:** JSON-RPC messages recoverable as structured text fragments within a memory image; no native on-disk format, since this artifact exists only when acquired from RAM.
- **Persistence/volatility:** Very high, this evidence exists only in memory and is lost on process termination or system shutdown; RAM acquisition must occur before the host application closes or the machine powers down.[48][48]
- **Collection considerations:** requires a memory-forensics workflow (e.g., a Volatility-based acquisition and analysis pipeline) rather than standard disk imaging; because this is a live-acquisition-only artifact, it should be prioritized during initial triage on a running, in-scope system rather than deferred to a later disk-only examination.[48][48]
- **Interpretation caution:** this is an emerging research area (published 2026) validated against a small number of specific agent/model combinations; treat findings from other agentic tools or MCP client versions as **[Likely]** rather than confirmed until independently verified.[48][48]
- **Evidentiary value:** High when recoverable, this is one of the only known techniques for reconstructing agentic tool-call content (as opposed to just execution metadata) when the agent or its host process does not persist logs to disk, directly relevant to Scope item 3, "remote AI tools or agentic systems interacting with the endpoint."

15. Opportunities for Timeline Reconstruction and Attribution

- Correlating Categories 1 (install), 2 (app content), 6 (browser content), 7 (execution traces), and 9 (network volume) against a common timeline is the most reliable path to a defensible finding, no single Windows artifact category should be presented standalone as proof of AI prompt/input activity.

Windows Artifact Summary Table

#	Category	Most probative artifact	Volatility	Evidentiary role
1	Installed app evidence	HKCU Uninstall keys, Event Log	Low	Corroborative (timeline anchor)
2	App logs/local DBs	Electron LevelDB (ChatGPT/Claude), LM Studio JSON, Ollama logs	Medium	**Highest, direct content**
3	Prompt/input traces	Clipboard History (ActivitiesCache.db)	High	Probative when present
4	Upload/download	Zone.Identifier ADS	Low	High when present

	evidence			
5	Clipboard/temp/autosave	%TEMP%, Office .asd	High	Corroborative
6	Browser artifacts	IndexedDB/LevelDB, Local Storage	Medium	**Very high, direct content**
7	OS execution traces	UserAssist, ShellBags, Jump Lists, Prefetch/AmCache	Low-Medium	Corroborative (timeline)
8	Persistence/services	Run keys, Scheduled Tasks	Low	Corroborative
9	Network traces	SRUM (SRUDB.dat)	Medium	Corroborative (volume/timing)
10	Search/index/staging	Windows.edb	Medium	High (deleted-content recovery)
11	Recall/OS AI snapshots	Recall snapshot store (Copilot+ only)	Medium	Corroborative, config-dependent
12	Output handling	Correlated from 3/4/7	Varies	Corroborative
13	Anti-forensic indicators	Absence + corroboration	N/A	Inference-level only
14	Memory forensics (agentic/MCP)	RAM-resident JSON-RPC traces	Very high	High when recoverable, live-acquisition only

macOS Findings

1. App Containers, Application Support, and Caches

- **What it may show:** local conversation caches, app configuration, session tokens, and Electron LevelDB stores for desktop AI apps (ChatGPT desktop, Claude desktop, Copilot app).[30][31][30][31] For Claude Desktop specifically, one documented investigation found that unsent chat drafts (text typed into the input box but not yet submitted) are cached in `~/Library/Application Support/Claude/IndexedDB/`, while Claude Desktop otherwise stores full conversation history server-side only, meaning the local desktop app itself does not cache complete past conversations for offline recovery.[41][41] Claude Code CLI sessions are a distinct, more persistent artifact: complete prompt/response transcripts are written as JSONL files under `~/.claude/projects/<project-slug>/*.jsonl`.[41][41]
- **Typical location(s):** `~/Library/Containers/<bundle-id>/Data/Library/Application Support/` (sandboxed apps); `~/Library/Application Support/<AppName>/` (non-sandboxed, including Claude Desktop's IndexedDB store and `~/.claude/projects/` for Claude Code); `~/Library/Caches/<bundle-id>/`.[25][30][31][41][25][30][31][41]
- **Expected data format:** LevelDB (Electron apps), SQLite, JSON configuration, plist.
- **Persistence/volatility:** Medium, Caches/ may be purged under storage pressure or explicit cache-clear; Application Support persists until uninstall or manual deletion.
- **Collection considerations:** sandboxed container paths require elevated/Full-Disk-Access-equivalent privileges for live collection (moot for dead-box/image acquisition); preserve directory structure and all three timestamp fields (mtime/atime/btime), since LevelDB's internal payload timestamps may differ from filesystem timestamps.[25][30][31][25][30][31]

- **Interpretation caution:** LevelDB write-ahead logs (`.log`) may retain stale/superseded records alongside current state (`.ldb`/`.sst`), do not assume the newest `.log` entry is authoritative without full LevelDB reconstruction.[14][14]
- **Evidentiary value:** High, direct conversation content when parseable; strongest evidence of specific AI tool usage and content.

2. Pasteboard/Clipboard Artifacts

- **What it may show:** text copied from AI output or into an AI input field; Universal Clipboard cross-device transfers.
- **Typical location(s):** no persistent native pasteboard history file by default (in-memory, managed by `pboard`/`cfprefsd`); if a third-party clipboard manager (Paste, Maccy, Raycast) is installed, check that app's own SQLite/plist store.[7][8][7][8]
- **Expected data format:** in-memory for native pasteboard; SQLite typically for third-party managers.[7][7]
- **Persistence/volatility:** Very high, lost on reboot or overwrite; capturable only via live memory acquisition or a present clipboard-manager app.
- **Collection considerations:** treat as volatile-evidence triage priority before shutdown, with capture time documented; inventory third-party clipboard-manager artifacts during app-container review.
- **Interpretation caution:** Universal Clipboard content may originate from a paired iOS/iPadOS device, do not attribute authorship without cross-device correlation.
- **Evidentiary value:** Medium-high when present; opportunistic, not reliable.

3. TCC.db (Transparency, Consent, and Control Database)

- **What it may show:** permission grants relevant to AI features, microphone access for voice-AI, Accessibility permissions for AI-assisted automation, Screen Recording grants for AI screen-analysis tools.
- **Typical location(s):** `~/Library/Application Support/com.apple.TCC/TCC.db` (per-user); `/Library/Application Support/com.apple.TCC/TCC.db` (system-wide).[9][9]
- **Expected data format:** SQLite.[9][9]
- **Persistence/volatility:** Low, persists until the permission is revoked or the app removed.
- **Collection considerations:** requires Full Disk Access to read live; straightforward on a forensic image. SIP may restrict live access even with elevated privileges, dead-box imaging is the reliable path.
- **Interpretation caution:** records the grant event and last-modified time, not every usage instance, proves capability/authorization at a point in time, not use on every occasion.
- **Evidentiary value:** Medium, contextual/corroborating, not direct content evidence.

4. Unified Logs

- **What it may show:** app launches, network activity indicators, in-app `os_log` events, and Apple Intelligence feature invocations (Writing Tools, Genmoji, etc.).
- **Typical location(s):** live query via `log show`/`log stream`; backing store at `/var/db/diagnostics/` (`Persist/`, `Special/`) plus `/var/db/uuidtext/`.[25][25]
- **Expected data format:** proprietary binary (`tracev3`), queried via the `log` CLI or parsed with third-party unified-log parsers.
- **Persistence/volatility:** Medium, `Persist/` logs are typically retained days-to-weeks depending on volume/storage pressure (not a fixed window); `Special/` (higher-priority) entries may persist longer.

- **Collection considerations:** collect ASAP, one of the more perishable non-memory artifacts; use `log collect` to bundle into a portable `.logarchive` early in triage.
- **Interpretation caution:** extremely high-volume and noisy, requiring targeted predicate filtering; absence of an expected entry does not prove absence of activity, it may reflect rotation/pruning.
- **Evidentiary value:** Medium-high for timeline/sequence of AI app or feature activity, especially correlated with other artifacts.

5. FSEvents

- **What it may show:** file-system-level create/modify/rename/delete history for AI app data directories, showing when conversation or export files changed even after content is deleted.
- **Typical location(s):** `/.fseventsd/` (per-volume, hidden at volume root).[25][25]
- **Expected data format:** proprietary binary, gzip-compressed event stream files; parsed with tools such as FSEventsParser.
- **Persistence/volatility:** Medium, retention depends on volume activity and free space; older files are pruned over time with no fixed window.
- **Collection considerations:** collect from every mounted volume in scope, including external/backup volumes; correlate the resulting event timeline against paths of interest from Category 1.
- **Interpretation caution:** records path-level events, not content, shows "modified," not what changed; use as a timeline skeleton and corroborate with content-bearing artifacts.
- **Evidentiary value:** High for timeline reconstruction and for proving deletion/anti-forensic activity even when the file itself is unrecoverable.

6. QuickLook Thumbnail Cache

- **What it may show:** thumbnail previews of files viewed/selected in Finder, potentially including AI-generated images or exported PDF conversation logs, sometimes surviving after the source file is deleted.
- **Typical location(s):** `~/Library/Group Containers/group.com.apple.quicklook/com.apple.quicklook.ThumbnailsAgent/thumbnails.db`.[26][27][28][26][27][28]
- **Expected data format:** SQLite database referencing separate thumbnail blob files or embedded blobs.
- **Persistence/volatility:** Medium, periodically pruned/vacuumed by the OS.
- **Collection considerations:** parse with a QuickLook-aware tool; cross-reference against FSEvents and the current file listing to identify "orphaned" thumbnails as candidate deleted-file evidence.[26][26]
- **Interpretation caution:** proves the file was previewed/selected in Finder, not necessarily opened and read; a downscaled thumbnail may be forensically insufficient to fully authenticate original content.
- **Evidentiary value:** Medium-high specifically for recovering visual evidence of deleted AI-generated images or screenshot-based chat exports.

7. Spotlight Metadata

- **What it may show:** indexed file metadata including `kMDItemWhereFroms` (showing a file was downloaded from an AI service's domain) and content-type classification.
- **Typical location(s):** `/.Spotlight-V100/` (per-volume, hidden); queried live via `mdls <file>` or `mdfind`.
- **Expected data format:** proprietary binary store; per-item attributes readable via `mdls`.
- **Persistence/volatility:** Low-medium, persists until reindexed (`mdutil -E`).

- **Collection considerations:** run `mdls` on candidate files identified via QuickLook cache or quarantine xattrs, a cheap, high-value corroboration step.
- **Interpretation caution:** `kMDItemWhereFroms` is populated by the downloading application (usually browsers, via quarantine metadata), its presence or absence depends on that app's behavior, not a universal guarantee.
- **Evidentiary value:** Medium-high, `kMDItemWhereFroms` is strong corroborating evidence linking a file to an AI service URL/domain.

8. Quarantine / Provenance Extended Attributes

- **What it may show:** the `com.apple.quarantine` extended attribute recording originating URL, download timestamp, and downloading app for internet-retrieved files (e.g., an AI-generated export downloaded via Safari/Chrome).
- **Typical location(s):** extended attribute on the file itself, read via `xattr -p com.apple.quarantine <file>` or `mdls`.
- **Expected data format:** structured string (flags; timestamp; agent; UUID); companion `kMDItemWhereFroms` is plist-encoded.
- **Persistence/volatility:** Low, persists with the file until explicitly cleared or moved to a filesystem that strips extended attributes.
- **Collection considerations:** always check xattrs on candidate AI-related exports; easy to miss if the imaging/triage tool does not surface extended attributes by default.
- **Interpretation caution:** set at download time by the downloading app (typically browsers), will not be present for files created directly by a desktop AI app's own export function unless that app also sets it; absence does not rule out AI origin.
- **Evidentiary value:** High when present, ties a file to a specific source URL and timestamp with strong evidentiary weight, a well-precedented artifact type in prior download-provenance cases.

9. Recent Items and Preference Property Lists

- **What it may show:** recently opened documents/apps, including AI apps launched and files opened through them.
- **Typical location(s):** `~/Library/Application Support/com.apple.sharedfilelist/` (system Recent Items, security-scoped bookmark format); per-app preference plists at `~/Library/Preferences/<bundle-id>.plist`.[24][25][24][25]
- **Expected data format:** binary/XML plist; sharedfilelist entries use bookmark data requiring bookmark-resolution parsing.
- **Persistence/volatility:** Medium, lists are typically capped in length (oldest entries roll off); the user can clear via "Clear Menu."
- **Collection considerations:** use a bookmark-data-aware parser, plain plist viewers only show opaque binary blobs.
- **Interpretation caution:** reflects last-opened time; the list can silently truncate older relevant entries, so absence does not mean an item was never opened.
- **Evidentiary value:** Medium, good corroboration for "user interacted with file X via app Y," weaker as standalone primary evidence.

10. Browser Artifacts Related to AI Web Interfaces (Safari, Chrome)

- **What it may show:** structured conversation history, draft prompts not yet submitted, session/auth tokens, and occasionally cached response payloads.
- **Typical location(s):** Safari: `~/Library/Safari/Databases/` and `~/Library/WebKit/<bundle-id>/WebsiteData/`; Chrome: `~/Library/Application Support/Google/Chrome/Default/IndexedDB/` and `Local Storage/leveldb/`.[10][12][13][14][15][10][12][13][14][15]
- **Expected data format:** Safari uses WebKit's own SQLite/blob hybrid storage; Chrome uses LevelDB for both IndexedDB and LocalStorage.
- **Persistence/volatility:** Medium, subject to browser storage-quota eviction, explicit "Clear Browsing Data," or low-disk-space auto-eviction.
- **Collection considerations:** the highest-value macOS artifact category for direct prompt/response content, prioritize early given eviction risk; requires purpose-built IndexedDB/LevelDB parsing, since generic SQLite viewers do not correctly decode LevelDB or WebKit's IndexedDB blob format.
- **Interpretation caution:** per-origin storage isolation means each vendor's data sits in a differently structured origin folder; a parser tuned for one vendor's schema will not parse another's without adaptation, and vendor front-end updates can silently change schema.
- **Evidentiary value:** Very high, the most direct source of actual prompt text and AI-generated response content.

11. Cloud-Synced or Containerized Application Storage (iCloud)

- **What it may show:** evidence that AI-app data, exported content, or configuration synced across a user's Apple devices.
- **Typical location(s):** `~/Library/Application Support/CloudDocs/` (iCloud Drive local cache); `~/Library/Containers/<bundle-id>/Data/Library/Application Support/CloudKit/`; sync daemon activity in Unified Logs (`bird`/`cloudd` subsystems).[25][25]
- **Expected data format:** native file formats within the CloudDocs mirror, plus CloudKit's own SQLite-based sync metadata.
- **Persistence/volatility:** Medium, local cache reflects sync state that can change asynchronously; presence/absence locally may lag or diverge from the cloud account.
- **Collection considerations:** treat local evidence as a sync-state snapshot at acquisition time, not authoritative for full account content, full account-level content typically requires legal process to the cloud provider, a separate acquisition path.
- **Interpretation caution:** do not conflate "found locally" with "originated on this device", synced content may have originated on a different Apple device; cross-device provenance needs additional corroboration.
- **Evidentiary value:** Medium, valuable for cross-device activity correlation, weaker for single-device attribution without corroboration.

12. Memory Forensics for Agentic and MCP (Model Context Protocol) Activity

- **What it may show:** the same class of volatile agentic/MCP tool-call evidence described under Windows Findings, Category 14, recoverable from macOS process memory rather than disk.
- **Typical location(s):** volatile process memory (RAM) of the host application (browser, IDE, or agent runtime); the same DFRWS USA 2026 study demonstrating this technique was conducted across platforms and is not Windows-specific.[48][48]

- **Expected data format:** JSON-RPC messages recoverable as structured text fragments within a memory image.
- **Persistence/volatility:** Very high, lost on process termination or shutdown; requires live memory acquisition.[48][48]
- **Collection considerations:** macOS memory acquisition has additional constraints relative to Windows (System Integrity Protection, kernel extension signing requirements for some acquisition tools) that should be resolved before an investigation requiring this artifact begins.
- **Interpretation caution:** emerging, narrowly validated research; treat as **[Likely]** for agent/tool combinations not directly tested.[48][48]
- **Evidentiary value:** High when recoverable; directly relevant to Scope item 3, "remote AI tools or agentic systems interacting with the endpoint."

macOS Artifact Summary Table

#	Category	Most probative artifact	Volatility	Evidentiary role
1	App containers/Application Support/Caches	Electron LevelDB stores	Medium	**High, direct content**
2	Pasteboard/clipboard	Live memory or 3rd-party manager DB	Very high	Opportunistic
3	TCC.db	Permission grant records	Low	Corroborative
4	Unified Logs	os_log AI-subsystem events	Medium	Corroborative (timeline)
5	FSEvents	Per-volume event stream	Medium	High (timeline/deletion proof)
6	QuickLook thumbnail cache	index.sqlite + thumbnails.data	Medium	Medium-high (visual recovery)
7	Spotlight metadata	kMDItemWhereFroms	Low-medium	Medium-high (corroborative)
8	Quarantine/provenance xattrs	com.apple.quarantine	Low	High when present
9	Recent Items/plists	sharedfilelist bookmarks	Medium	Medium (corroborative)
10	Browser artifacts	Safari WebKit storage, Chrome LevelDB	Medium	**Very high, direct content**
11	iCloud/container sync	CloudDocs cache, CloudKit metadata	Medium	Medium (cross-device correlation)
12	Memory forensics (agentic/MCP)	RAM-resident JSON-RPC traces	Very high	High when recoverable, live-acquisition only

Comparative Analysis: Windows vs. macOS

The same user behavior, typing a prompt, pasting sensitive text, uploading a file, or downloading an AI-generated output, leaves materially different categories of evidence on Windows and macOS.[16][25][16][25] An investigator who applies a Windows-first mental model to a Mac (or vice versa) risks missing the platform's actual highest-value artifact.

Where the platforms are similar: both platforms converge on the same conclusion for the single most probative artifact family, browser and desktop-app local storage (IndexedDB/LevelDB), because both Chromium (Windows

and macOS) and the Electron framework used by most desktop AI apps rely on the same underlying storage engine regardless of OS.[10][14][10][14] A parser built for one platform's LevelDB schema is largely portable to the other, which is a key design implication in Proposed Tool Design. The two platforms also converge on memory forensics as the recovery path for agentic/MCP tool-call activity (Windows Findings, Category 14; macOS Findings, Category 12): the underlying JSON-RPC message structure and the requirement for live RAM acquisition are the same on both operating systems, only the acquisition tooling and platform-security constraints differ.[48][48]

Where the platforms differ:

Dimension	Windows	macOS
Native clipboard history	Explicit opt-in feature (Clipboard History), persisted to a SQLite database when enabled[1][2][1][2]	No native persistent pasteboard history; recoverable only via live memory or a third-party clipboard manager
Execution/access timeline artifacts	Rich registry-based ecosystem: UserAssist, ShellBags, Prefetch, AmCache, Jump Lists[16][17][18][19][16][17][18][19]	Primarily Unified Logs and FSEvents; no direct registry equivalent
Download provenance	Zone.Identifier alternate data stream, well-precedented in prior case law and tooling[37][38][37][38]	Quarantine extended attribute + kMDItemWhereFroms Spotlight metadata, functionally similar but structurally distinct
Deleted-content recovery	Windows Search index (Windows.edb) can retain fragments of deleted, indexed document text[35][36][35][36]	QuickLook thumbnail cache can retain visual fragments of deleted files; no direct text-content equivalent
OS-native AI feature snapshot	Windows Recall (Copilot+ hardware only, opt-in)	No direct equivalent at time of writing
Permission/capability history	No single unified permission database equivalent to TCC	TCC.db provides a durable, centralized permission-grant history
Network volume/timing	SRUM (SRUDB.dat), per-application, per-time-bucket network usage	No documented direct equivalent with comparable per-app timing granularity

Practical implication: an investigator should not expect a one-to-one artifact mapping across platforms.[9][32][9][32] A Windows-only artifact (SRUM, Zone.Identifier) can be a critical corroborating link that simply has no macOS analog, and vice versa (TCC.db, QuickLook cache). Cross-reference this table against the Windows Findings and macOS Findings sections when scoping a mixed-fleet or BYOD investigation, and do not treat the absence of a platform-specific artifact on the "wrong" OS as a gap in the examination, it may be an expected structural difference.

Proposed Tool Design: "PromptTrace" (Illustrative)

This is a fictional/illustrative product design for white-paper and planning purposes, not a validated, released, or court-tested tool, and not legal or forensic-admissibility guidance. It is detailed enough that a developer could build a proof of concept from it.

Purpose and Target Users

A cross-platform (Windows + macOS) forensic utility purpose-built to discover, collect, parse, normalize, and report on AI prompt/input and response activity artifacts across browser, desktop-app, and OS-native evidence

sources, producing an examiner-reviewable, defensible artifact report rather than raw data dumps.[10][14][32][10][14][32] Target users are digital forensic examiners (law enforcement, corporate/eDiscovery, incident response) triaging systems suspected of relevant AI tool usage, and forensic tool developers seeking a reference architecture to integrate into an existing suite.

Supported Evidence Categories (v1 scope)

1. Browser AI-web-UI storage (IndexedDB/LevelDB/LocalStorage), Chrome, Edge, Safari, Firefox.[10][13][14][10][13][14]
2. Desktop AI app containers (ChatGPT, Claude, Copilot desktop apps), Electron LevelDB + SQLite.
3. AI coding-assistant local storage (editor/IDE extension storage).
4. OS-native AI feature logs (Windows Recall store where present, Copilot interaction logs; macOS Unified Log AI-subsystem events, TCC grants).[9][25][33][9][25][33]
5. Supporting timeline artifacts (FSEvents on macOS, Prefetch/AmCache/UserAssist/BAM on Windows, QuickLook on macOS), corroboration, not standalone content extraction.[16][25][26][44][16][25][26][44]
6. Provenance metadata (quarantine xattrs on macOS, Zone.Identifier ADS on Windows, kMDItemWhereFroms).[37][38][37][38]
7. Volatile agentic/MCP tool-call memory artifacts (Windows Findings, Category 14; macOS Findings, Category 12), recoverable only via live memory acquisition.[48][48]

Collection Workflow

8. **Triage/Scoping**, the examiner selects a target (live system, mounted image, or evidence file) and records case metadata (case number, examiner ID, evidence item ID) to a manifest before any collection begins.[9][9]
9. **Volatile capture (live systems only, optional)**, clipboard/pasteboard state and the running-process list are captured first as the highest-volatility items, flagged distinctly as "live-volatile" with a precise capture timestamp.[3][7][3][7]
10. **Targeted artifact acquisition**, a config-driven, per-OS/per-browser/per-app path manifest is walked, copying relevant files/directories with original timestamps preserved.
11. **Hashing**, a SHA-256 hash of every acquired file is logged to the collection manifest immediately after each copy, not batched at the end, so a partial or interrupted collection still has integrity records.
12. **Parsing (offline, on collected copies only)**, each artifact type is routed to its dedicated parser module; the tool never touches source evidence again after acquisition.
13. **Normalization and correlation**, parsed records are mapped into a common schema and timeline-correlated.
14. **Reporting**, a structured report plus a raw-data export is generated for independent verification.

Artifact Normalization Strategy

A single intermediate record schema is used regardless of source, so the correlation/timeline engine and report generator work identically across LevelDB, plist, SQLite, and binary-log formats:

```
{
  artifact_id, source_category (browser|desktop_app|os_native|provenance|timeline),
  os, application/vendor, origin_path,
  event_type (prompt_submitted|response_received|app_launch|permission_grant|file_created|...),
  timestamp (explicit timezone + source-clock caveat), timestamp_confidence (high|medium|low),
  content_excerpt (nullable, populated only when full text recovery achieved),
  content_recovery_status (full|partial|metadata_only|unrecoverable),
  parser_used, parser_version, schema_version_detected,
```

```
  raw_source_reference (pointer back to original collected file + byte offset where applicable)
}
```

The `raw_source_reference` field keeps every normalized record traceable back to its original evidentiary source for verification or challenge.

Parsing and Reporting Approach

A modular parser-plugin architecture is used: each artifact type (for example, "ChatGPT web IndexedDB schema v3") is an independent plugin with its own version number, so a vendor schema change requires only a plugin update, not a core-engine rewrite.[14][39][14][39] Plugins declare their own schema-detection signature (expected LevelDB key prefixes, SQLite table names) and self-report a confidence of correct version match, feeding the confidence-labeling scheme below. Parsing runs read-only against copies only, in an isolated working directory. The reporting layer is fully separated from parsing, parsers emit normalized JSON, and the report generator consumes only that schema, so report templates can evolve independently of parser internals.

Confidence Labeling Scheme

Each normalized record carries a combined confidence label from three factors:

- **Schema Match Confidence:** High (exact known schema matched) / Medium (close match, minor unexpected fields) / Low (heuristic best-effort parse of an unrecognized variant).
- **Content Completeness:** Full / Partial (e.g., a LevelDB compaction left a fragment) / Metadata-only (existence and timestamp known, content not recoverable).
- **Timestamp Reliability:** High (artifact's own timestamp, cross-validated against filesystem metadata) / Medium (single-source, unvalidated) / Low (inferred/approximate, e.g., derived from surrounding FSEvents).

The combined label, e.g., "[High schema / Full content / High timestamp]", surfaces in the report per finding, so examiners and opposing experts can gauge evidentiary weight immediately; a full methodology appendix remains available for challenge.

Handling of Unsupported or Unknown Artifacts

Any file or store matched to a known artifact path but not successfully parsed is flagged "Recognized-but-unsupported" and preserved in full (hash, path, file type) in a dedicated report appendix, never silently dropped.[9][9] Any artifact discovered outside the known-path manifest but heuristically resembling AI-related storage is flagged "Candidate/unclassified" for manual examiner review; the design deliberately errs toward surfacing more candidates, since silent omission is the higher-risk failure mode. A versioned, documented "known schema" changelog lets examiners distinguish a tool limitation at exam time from a genuinely absent artifact.

Chain-of-Custody and Forensic-Soundness Considerations

Write-blocking or read-only mounting is enforced for all image-based collection; live-system collection is explicitly flagged as inherently altering some system state (e.g., access-time updates) and documented as such rather than claimed forensically pristine.[9][9] Every acquisition action is logged to an append-only collection log (tool version, timestamp, examiner, source path, resulting hash), and that log is itself hashed and included in the final report package.[9][9] The tool never writes back to source evidence under any code path, enforced architecturally (e.g., OS-level read-only mount in addition to application-level safeguards). Standard image formats (E01/Ex01, AFF4, raw/dd) are supported via existing libraries rather than reinventing image parsing.

Privacy and Legal Sensitivity Notes

AI conversation content frequently includes highly sensitive personal, medical, financial, or privileged material.[10][14][10][14] The default export/report behavior redacts or truncates content excerpts unless the examiner explicitly enables full-content export, reducing inadvertent overexposure in draft reports circulating before legal review. Where privilege review is relevant (e.g., corporate legal matters), the workflow provides a hook for a privilege-review pass before final report distribution.[9][9] Verbatim recovered artifact content is always visually distinguished from tool-inferred or interpreted content (a "likely AI-generated" heuristic flag carries fundamentally different evidentiary weight than a directly recovered prompt string). Jurisdictional variation in what is collectible or admissible is documented explicitly, the tool provides technical capability, not a legal-sufficiency determination, and reports carry a standard disclaimer to that effect.

Report Structure

15. Case/examiner metadata, tool version, and collection-manifest hash summary.
16. Executive summary (plain language): what AI activity was found, on what dates, with what confidence.
17. Findings by evidence category, each tagged with its confidence label.
18. Timeline view: chronological correlation across all evidence categories (e.g., "10:03, ChatGPT web prompt submitted [browser IndexedDB]; 10:04, file created in Downloads [FSEvents]; 10:04, quarantine xattr shows origin chatgpt.com [xattr]").
19. Recognized-but-unsupported and Candidate/unclassified appendices, for full transparency on gaps.
20. Methodology appendix: parser versions, schema-match logic, known limitations.
21. Raw-data export package (hashes plus normalized JSON) for independent re-verification by a second examiner or opposing expert.
22. Standard disclaimers: forensic-soundness caveats for live collection, jurisdictional/admissibility disclaimer, redaction notice if applied.

Future Extensibility

A plugin SDK allows third parties or examiners to contribute new vendor-schema parsers without core-engine changes, important given how quickly AI vendor storage schemas change.[14][39][14][39] A mobile-extension path (iOS/Android AI app and browser artifacts) is a natural v2 target, reusing the same normalized schema and confidence-labeling model. A cloud-provider legal-process response ingestion module could merge subpoenaed cloud-side data into the same timeline/report framework. Automated schema-drift detection, periodic canary testing against live AI web UIs in a lab environment, would catch vendor storage-format changes proactively rather than discovering breakage mid-case.

Figure 2 (see Appendix A, Figure Specifications) illustrates this workflow as a four-stage pipeline: Collection → Normalization → Confidence Scoring → Reporting, framed by an Evidence Sources band and a Preservation and Audit Controls band.

Capabilities and Limitations

What this paper's findings support today: direct recovery of prompt and response text from browser and Electron-based desktop-app LevelDB/IndexedDB stores when not yet evicted;[10][14][10][14] recovery of file-download provenance via Zone.Identifier (Windows) or quarantine xattrs/kMDItemWhereFroms (macOS);[37][38][37][38] timeline correlation of application execution and network activity via OS-native artifacts;

and visual recovery of deleted-file evidence via QuickLook thumbnails (macOS) or Windows Search index fragments (Windows).[26][35][26][35]

What requires caution or further validation: exact recoverability varies by AI vendor, application version, and browser version, and is not guaranteed, LevelDB compaction, storage-quota eviction, and vendor schema changes can each independently remove or reshape the underlying data;[14][39][14][39] Windows Recall availability is hardware- and configuration-dependent and should never be assumed present; and local-LLM artifact locations (Ollama, LM Studio) are drawn from a single systematic academic study at time of writing and should be independently re-verified against the installed version before being relied upon in casework.[39][40][39][40]

Encryption and Protected App Containers

Full-disk encryption, sandboxed app containers requiring elevated privileges, and vendor-side end-to-end encryption of cloud-synced content can each place relevant data outside the reach of local acquisition.[9][25][9][25] When encountered, document the barrier explicitly rather than reporting an artifact as absent.

Ephemeral Data Loss

Clipboard/pasteboard content, LevelDB write-ahead-log fragments prior to compaction, and browser session-restore state are all subject to loss on reboot, application restart, or routine storage management, independent of any user action.[4][7][14][4][7][14] Collection timing materially affects what is recoverable.

Browser and Session Limitations

Private/incognito browsing sessions are designed to minimize persistent local storage; multi-profile browser configurations can fragment evidence across several origin folders; and browser storage-quota eviction can remove older IndexedDB/LocalStorage records without any user-initiated deletion.[10][13][14][10][13][14]

Absence of Direct Proof of Exact Submitted Text

Even where an artifact confirms that an AI application or web interface was used, the exact text submitted in a given interaction may not be recoverable, particularly when the client discards it after sending and cloud-side logs are the only durable copy.[11][20][21][11][20][21] Where local artifacts do not capture full content, this paper distinguishes "the interface was used" from "this exact text was submitted," and investigators should preserve that distinction in their own reporting.

Ambiguity Between AI Use and Non-AI Use

A browser visit to an AI vendor's domain, or a file with no distinguishing origin metadata, does not by itself establish that AI-specific interaction occurred versus, for example, viewing marketing content on the same domain.[13][13] Corroborate domain-level evidence with content-bearing artifacts before concluding AI interaction took place.

Attribution Limitations

Artifacts tied to a user profile or account are not automatically proof of the specific individual who typed the content, shared accounts, unattended sessions, and synced multi-device activity (Universal Clipboard, iCloud) all complicate attribution and should be corroborated with additional evidence (login timing, device presence, physical access records) before being presented as person-level attribution.[8][25][8][25]

Version and Platform Differences

Every specific file path, schema, and artifact behavior described in this paper is tied to the application and OS versions current at the time the underlying source was published or the research was conducted, and should be independently re-verified at exam time rather than assumed stable, this is explicitly true for AI desktop applications and browser extensions, which update on vendor timelines far faster than traditional forensic artifacts.[39][39]

Anti-Forensic and Privacy Features

Native OS privacy features (private browsing, Clipboard History being off by default, Recall being opt-in) and user-driven clearing of history or caches can each independently reduce available evidence without implying deliberate anti-forensic intent[1][33][1][33], see the caution in Windows Findings, Category 13.

Gaps Requiring Future Research

Reliable, vendor-neutral parsing of Electron/Chromium LevelDB stores across rapid schema churn; systematic (not single-study) validation of local-LLM artifact locations across more tools and versions; a documented, stable set of macOS artifacts for OS-native AI features as Apple Intelligence capabilities expand; empirical testing of Windows Recall's actual forensic recoverability once broader hardware deployment allows controlled study; and broader validation of memory-forensic MCP/agentic tool-call reconstruction beyond the specific agent/model combinations tested in the single DFRWS USA 2026 study currently available.[39][40][48][39][40][48]

Investigator Workflow Recommendations

23. **Scope first.** Confirm which AI applications, browsers, and OS features are plausibly in play before collection, using Installed AI Applications evidence (Windows Findings, Category 1; macOS Findings, Category 1) as the starting inventory.[5][32][5][32]
24. **Collect volatile evidence first.** Clipboard/pasteboard state, if in scope, should be captured before any action that could trigger a reboot or application restart.[3][7][3][7]
25. **Prioritize browser and desktop-app local storage.** This is the highest-probative-value category on both platforms (Windows Findings, Category 2 and 6; macOS Findings, Category 1 and 10) and the most vulnerable to ongoing eviction, acquire it early.[10][14][10][14]
26. **Build the timeline from multiple corroborating categories.** No single artifact category should be presented standalone as proof of AI prompt/input activity (see Comparative Analysis and Windows Findings, Category 14).
27. **Document absence explicitly.** When an expected artifact is missing, record the plausible reasons (feature disabled, eviction, version difference) rather than treating absence as conclusive of anything.
28. **Separate confirmed content recovery from inference.** Apply the three-tier evidentiary standard from Methodology and Evidence Standard consistently in the final report.
29. **Flag jurisdiction- and privilege-sensitive content early**, per the Privacy and Legal Sensitivity Notes in Proposed Tool Design, before broader report circulation.

Conclusion

Determining whether a user submitted sensitive content to an AI system is achievable with defensible evidence today, but only by correlating multiple artifact categories rather than relying on any single

source.[10][11][14][10][11][14] Browser and desktop-app local storage (IndexedDB/LevelDB) is the evidentiary core on both Windows and macOS; OS-native execution, provenance, and permission artifacts provide the corroborating timeline and context that make a finding defensible rather than speculative.[10][14][37][10][14][37] The two platforms diverge meaningfully in which corroborating artifacts are available, Windows offers a richer native execution-history ecosystem and per-application network-volume data, while macOS offers a centralized permission history and stronger visual-recovery options for deleted files, so investigators and tool developers alike should build platform-aware, not platform-uniform, collection and analysis workflows.[9][16][32][9][16][32] The PromptTrace tool design in this paper illustrates one way to operationalize that correlation discipline into a repeatable, examiner-reviewable process; a developer could use its normalization schema, confidence-labeling scheme, and modular parser architecture as a concrete starting point for a proof of concept.

Appendices

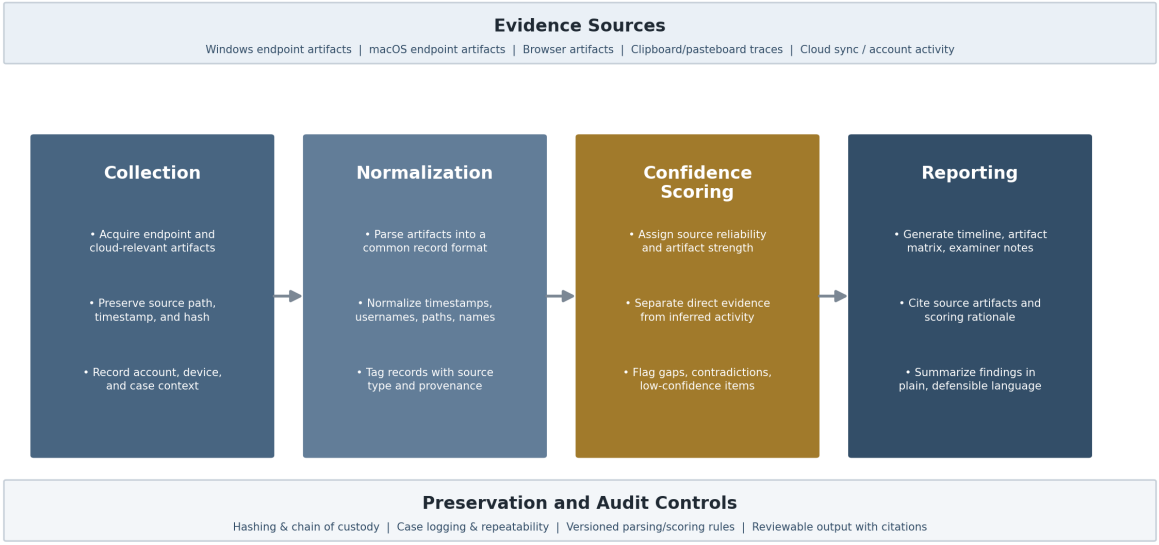
Appendix A. Figure Specifications

Figure 1. Windows and macOS Artifact Comparison for AI Prompt/Input Activity

Evidence category	Windows examples	macOS examples
Installed app evidence	Installed Programs / Apps & Features Program Files or AppData\Local\<vendor> Prefetch / execution traces	/Applications ~/Library/Application Support/<vendor> LaunchServices / quarantine metadata
Browser artifacts	Browser profile folders History, Cookies, Login Data Local Storage, downloads, cache	Browser profile folders History, Cookies, Login Data Local Storage, downloads, cache
Clipboard	Clipboard History (if enabled) Recent copy/paste traces App-specific paste buffers	Pasteboard activity (in-memory) App-specific paste traces Clipboard-manager artifacts
OS traces	Recent files, Jump Lists LNK files, ShellBags, event logs Temp files, user-profile artifacts	Recent Items App support files, logs, plist artifacts Temp files, user-profile artifacts
Cloud sync	OneDrive / Microsoft account sync Browser sync, vendor account activity Cloud-side audit / session logs	iCloud Drive / iCloud sync Browser sync, vendor account activity Cloud-side audit / session logs

Examples are indicative, not exhaustive; artifact locations vary by application version, user profile, and synchronization settings.

Figure 2. Proposed Forensic Tool Architecture for AI Prompt/Input Evidence



A defensible workflow for collecting, normalizing, scoring, and reporting AI prompt/input artifacts from Windows and macOS systems.

Figure 1. Windows and macOS Artifact Comparison for AI Prompt/Input Activity

Caption: Parallel evidence categories and representative artifacts for Windows and macOS systems that may retain prompt/input activity traces. A 3-column table (Evidence category / Windows examples / macOS examples) covering Installed app evidence, Browser artifacts, Clipboard, OS traces, and Cloud sync, with the note: "Examples are indicative, not exhaustive; artifact locations vary by application version, user profile, and synchronization settings." (See the Comparative Analysis table above for the full text content.)

Figure 2. Proposed Forensic Tool Architecture for AI Prompt/Input Evidence

Caption: A defensible workflow for collecting, normalizing, scoring, and reporting AI prompt/input artifacts from Windows and macOS systems. A horizontal four-stage pipeline (Collection → Normalization → Confidence Scoring → Reporting) framed by an "Evidence Sources" band above and a "Preservation and Audit Controls" band below, with a "Primary outputs" callout (Timeline, Artifact matrix, Confidence summary, Source appendix). (See Proposed Tool Design above for full stage content.)

Both figures use a professional forensic palette (navy/slate/muted blue/amber accents; no gradients, no marketing gloss) and are sized to fit a single Word page width, per graphics specification.

Appendix B. Artifact Matrix by Platform

Platform	Category	Artifact	Format
Windows	Installed app evidence	HKCU Uninstall keys, Event Log	Registry, .evtx
Windows	App logs/local DBs	Electron LevelDB, LM Studio JSON, Ollama server.log	LevelDB, JSON, text
Windows	Prompt/input traces	ActivitiesCache.db (Clipboard History)	SQLite
Windows	Upload/download evidence	Zone.Identifier ADS	NTFS ADS text
Windows	Browser artifacts	IndexedDB/LevelDB, Local	LevelDB

		Storage	
Windows	OS execution traces	UserAssist, ShellBags, Jump Lists, Prefetch, AmCache	Registry, binary
Windows	Network traces	SRUDB.dat	ESE database
Windows	Search/staging	Windows.edb	ESE database
Windows	OS AI snapshots	Recall snapshot store (Copilot+ only)	Proprietary
Windows	Memory forensics (agentic/MCP)	RAM-resident JSON-RPC traces	Volatile memory image
macOS	App containers	Electron LevelDB, SQLite, plist	LevelDB, SQLite, plist
macOS	Pasteboard	In-memory / 3rd-party manager DB	SQLite (3rd-party)
macOS	Permissions	TCC.db	SQLite
macOS	Unified Logs	tracev3 logarchive	Proprietary binary
macOS	File-system timeline	FSEvents stream	Proprietary binary
macOS	Visual recovery	QuickLook thumbnails.db	SQLite + blob
macOS	Metadata	Spotlight (mdls/mdfind)	Proprietary binary
macOS	Provenance	com.apple.quarantine xattr	Extended attribute
macOS	Recent items	sharedfilelist, plist	Binary/XML plist
macOS	Browser artifacts	Safari WebKit storage, Chrome LevelDB	SQLite/blob, LevelDB
macOS	Cloud sync	CloudDocs cache, CloudKit metadata	Native files, SQLite
macOS	Memory forensics (agentic/MCP)	RAM-resident JSON-RPC traces	Volatile memory image

Appendix C. Evidentiary Value Ranking (Highest to Lowest)

30. Browser/desktop-app IndexedDB/LevelDB stores (both platforms), direct prompt/response content.[10][14][10][14]
31. Download provenance (Zone.Identifier / quarantine xattr + kMDItemWhereFroms), high when present.[37][38][37][38]
32. Windows Search index (Windows.edb) and macOS QuickLook cache, deleted-content/visual recovery.[26][35][26][35]
33. TCC.db (macOS) and SRUM (Windows), durable, high-value corroboration.[9][32][9][32]
34. OS execution-timeline artifacts (UserAssist/ShellBags/Prefetch/AmCache; Unified Logs/FSEvents), corroborative timeline.[16][17][25][16][17][25]
35. Clipboard/pasteboard artifacts, probative when present, but frequently unavailable.
36. Recent items, persistence/service artifacts, Recall snapshots, contextual/corroborative, configuration-dependent.
37. Volatile agentic/MCP memory artifacts, high value when recoverable, but recovery depends entirely on live acquisition timing and is validated only for specific tested agent/tool combinations at time of writing.[48][48]

Appendix D. Volatility Ranking (Most to Least Volatile)

38. Volatile agentic/MCP tool-call memory artifacts, RAM-only, lost on process termination or shutdown.[48][48]
39. Native clipboard/pasteboard content (not yet in a history feature), memory-only.[3][7][3][7]
40. LevelDB write-ahead-log entries prior to compaction.
41. Browser IndexedDB/LocalStorage subject to storage-quota eviction.[10][14][10][14]
42. Temp/autosave files (%TEMP%, saved-application-state).

43. Unified Logs `Persist/` store and FSEvents (days-to-weeks, pruned).[25][25]
44. Windows Search index and QuickLook thumbnail cache (periodically pruned/vacuumed).[26][35][26][35]
45. Registry-based execution artifacts (UserAssist, ShellBags, AmCache) and TCC.db, durable until explicit action.[9][16][9][16]
46. Quarantine/provenance extended attributes and Zone.Identifier ADS, persist with the file.[37][37]
47. Installed-application evidence (uninstall keys), persists until uninstall.

Appendix E. Collection Priority Recommendations

48. Volatile agentic/MCP tool-call memory artifacts (live systems only, where agentic/MCP tool use is in scope), before process termination.[48][48]
49. Volatile clipboard/pasteboard state (live systems only, before any shutdown risk).
50. Browser and desktop-app IndexedDB/LevelDB stores (both platforms), before storage-quota eviction or app cache-clear.
51. Unified Logs (`log collect`) and FSEvents on macOS, before rotation/pruning.
52. Download-provenance metadata (Zone.Identifier, quarantine xattrs, Spotlight), low volatility, but easy to miss without ADS/xattr-aware tooling.
53. OS execution-timeline artifacts (UserAssist, ShellBags, Prefetch, AmCache, TCC.db), durable, lower time pressure.
54. Network-volume artifacts (SRUM) and search-index artifacts (Windows.edb, QuickLook cache), durable but benefit from timely ESE recovery/repair.
55. Installed-application and persistence evidence, least time-sensitive.

Appendix F. Candidate Test Scenarios for Validation

56. Submit a known, unique prompt string via a Chromium browser to a cloud AI web UI; image the system within 5 minutes and again after 24 hours; compare IndexedDB recoverability.
57. Install and use a desktop AI app (Electron-based); force-quit vs. graceful-quit the application; compare LevelDB `.log`/`.ldb` state in each case.
58. Enable Windows Clipboard History, copy a known string into an AI web UI's input field, and verify recoverability in `ActivitiesCache.db`.
59. Download a known AI-generated file via Safari and via Chrome on the same macOS system; compare quarantine xattr and `kMDItemWhereFroms` population between browsers.
60. Install Ollama and LM Studio on a clean Windows VM; run a known prompt through each; verify artifact paths and content against the locations documented in this paper.[39][40][39][40]
61. Enable Windows Recall on eligible hardware, display a known AI chat session on screen, and assess snapshot/OCR recoverability of that content.
62. Delete a file after generating a QuickLook thumbnail (macOS) or after Windows Search indexing (Windows); assess recoverability of the deleted content via each platform's respective mechanism.
63. Acquire a memory image from a system actively running an MCP-based agentic tool mid-session; attempt to reconstruct tool-invocation JSON-RPC traces and compare recoverability against the tested tool/model combinations reported in the source study.[48][48]

Appendix G. Glossary

- **Artifact**, any recoverable trace of activity on a system.
- **Indicator**, an observation that suggests, but does not confirm, that an event occurred.

- **Prompt**, user-submitted input to an AI system.
- **Completion / response**, AI-generated output content returned to the user.
- **Volatility**, the likelihood that an artifact will be lost, overwritten, or altered over time or by routine system activity.
- **Probative**, evidence that directly supports a specific factual conclusion (e.g., recovered prompt text).
- **Corroborative**, evidence that supports or strengthens a conclusion drawn primarily from other evidence, but does not independently establish it.
- **Chain of custody**, the documented, unbroken record of who collected, handled, and analyzed evidence, and when.
- **Confidence labeling**, a structured indication (e.g., schema match, content completeness, timestamp reliability) of how much weight a given normalized finding should carry.
- **LevelDB**, a key-value storage engine used by Chromium-based browsers and many Electron desktop applications for IndexedDB and Local Storage.
- **Extended attribute (xattr)**, metadata attached to a file outside its primary data stream, used on macOS for quarantine/provenance tracking.
- **Alternate Data Stream (ADS)**, an NTFS feature allowing additional named data streams attached to a file, used on Windows for the Zone.Identifier download-provenance marker.
- **Model Context Protocol (MCP)**, a JSON-RPC-based protocol that lets AI agents and coding assistants discover and invoke external tools; MCP tool-call traffic is a distinct forensic target from prompt/response content because it can reveal what actions an agent took, not just what it was asked.

Companion Memo: Research Gaps, Assumptions, and Recommended Next Steps for Tool Prototyping

Prepared for: technical decision-makers evaluating a PromptTrace-style proof of concept.

Key assumptions carried through this paper:

- The investigator has appropriate legal authority to examine the system in question.
- Artifact paths and schemas cited are version-dependent and must be re-verified against the specific installed application/OS build at exam time.[39][39]
- Local-LLM artifact findings (Ollama, LM Studio) rest on a single systematic academic study at time of writing and have not yet been cross-validated by a second independent research effort.[39][40][39][40]
- Windows Recall's forensic recoverability characteristics are inferred from its documented design and are not yet supported by a dedicated forensic validation study at time of writing.
- Memory-forensic reconstruction of agentic/MCP tool-call activity (Windows Findings, Category 14; macOS Findings, Category 12) rests on one DFRWS USA 2026 study tested against a small set of specific agent/model combinations and should not be assumed to generalize to every MCP client or agentic framework.[48][48]

Research gaps to close before a production tool relies on this paper's findings:

64. Independent, repeatable validation of Electron/Chromium LevelDB parsing across a wider set of AI vendor front-end versions than currently published.[14][14]
65. A dedicated forensic study of Windows Recall snapshot recoverability once broader Copilot+ hardware deployment permits controlled testing.
66. Expanded macOS Apple Intelligence artifact documentation as OS-native AI features expand beyond Writing Tools and Genmoji.
67. A second independent study validating local-LLM (Ollama, LM Studio, llama.cpp) artifact locations across additional versions and configurations.[39][40][39][40]
68. Empirical eviction-timing studies for browser IndexedDB/LocalStorage under realistic usage patterns, to give investigators a defensible collection-window estimate rather than a qualitative "collect early" guideline.
69. Independent replication of the memory-forensic MCP/agentic tool-call reconstruction technique against additional agent frameworks and MCP client versions beyond those tested in the source study.[48][48]

Recommended next steps for tool prototyping:

70. Build the LevelDB/IndexedDB parser module first, it addresses the single highest-value, most time-sensitive artifact category on both platforms.
71. Validate the normalized JSON schema and confidence-labeling scheme against the test scenarios in Appendix F before building the full plugin architecture.
72. Prioritize Windows (Zone.Identifier, UserAssist/ShellBags/Prefetch/AmCache) and macOS (quarantine xattr, TCC.db, FSEvents) parser plugins next, given their durability and established forensic precedent.
73. Defer Windows Recall and Apple Intelligence-specific parsing until dedicated validation research is available, to avoid building against an unstable or unvalidated target.
74. Pilot the tool against the test scenarios in Appendix F on lab systems before any casework use, and document parser-version-to-schema-version mappings from day one to support the "Recognized-but-unsupported" reporting category.

75. Treat memory-forensic MCP/agentic reconstruction (Appendix F, scenario 8) as a stretch goal for a later release phase rather than v1 scope, given its narrow validation base at time of writing.[48][48]

Open questions for reviewer/stakeholder confirmation:

- Should v1 scope include mobile-device artifacts, or remain strictly desktop/laptop as scoped in this paper?
- What is the organization's position on collecting and reporting verbatim prompt content by default versus requiring an explicit override, given the privacy sensitivity noted in Proposed Tool Design?
- Is there an internal or partner lab environment available to run Appendix F's test scenarios under controlled, version-documented conditions?

References

[1] Lian (Inversecos), "How to Perform Clipboard Forensics: ActivitiesCache.db, Memory Forensics and Clipboard History," *Inversecos,* https://www.inversecos.com/2022/05/how-to-perform-clipboard-forensics.html, May 2022, Last accessed September 18th 2026

[2] Insider Threat Matrix, "Clipboard Payloads via ActivitiesCache.db (Detection DT090)," *Insider Threat Matrix Knowledge Base,* https://insiderthreatmatrix.org/detections/DT090, 2024, Last accessed September 18th 2026

[3] A. Savoldi and P. Gubian, "Extracting the Windows clipboard from physical memory," *Digital Investigation, vol. 8, pp. S118–S124,* https://www.researchgate.net/publication/251702413_Extracting_the_windows_clipboard_from_physical_memory, 2011, Last accessed September 18th 2026

[4] Ret2pwn, "The Art of Clipboard Forensics: Recovering Deleted Data," *RET2Pwn Blog,* https://xret2pwn.github.io/The-Art-of-Clipboard-Forensics-Recovering-Deleted-Data, 2023, Last accessed September 18th 2026

[5] Magnet Forensics, "Computer Artifacts: Exploring Metadata, Log Files, Registry Data, and More," *Magnet Forensics Blog,* https://www.magnetforensics.com/blog/computer-artifacts-exploring-metadata-log-files-registry-data-and-more, July 2024, Last accessed September 18th 2026

[6] Kevia Oh, "Clippy History (Windows Clipboard History forensic notes)," *ThinkDFIR,* https://thinkdfir.com/2018/10/14/clippy-history, October 14th, 2018, Last accessed September 18th 2026

[7] NSPasteboard.org, "Identifying and Handling Transient or Special Data on the Clipboard," *NSPasteboard.org,* http://nspasteboard.org, 2018, Last accessed September 18th 2026

[8] Alok Tiwari, "macOS Clipboard History — What Gets Logged, What Leaks, and How to Audit It," *DEV Community,* https://dev.to/alokt/macos-clipboard-history-what-gets-logged-what-leaks-and-how-to-audit-it-4knj, August 2024, Last accessed September 18th 2026

[9] Magnet Forensics, "7 Essential Artifacts for macOS Forensics," *Magnet Forensics Blog,* https://www.magnetforensics.com/blog/essential-artifacts-for-macos-forensics, September 2024, Last accessed September 18th 2026

[10] J. Han and T. Shon, "Forensic analysis of browser-based conversational AI assistant extensions: Monica, Sider, and MaxAI," *Forensic Science International: Digital Investigation, vol. 56, p. 301934,* https://www.sciencedirect.com/science/article/abs/pii/S2666281726000909, 2026, Last accessed September 18th 2026

[11] M. W. Kankanamge, N. McKenna, S. Challa, and A. Imteaj, "Digital Forensic Investigation of the ChatGPT Windows Application," *arXiv preprint arXiv:2505.23938,* https://arxiv.org/html/2505.23938v1, May 2025, Last accessed September 18th 2026

[12] F. Paligu and C. Varol, "Browser Forensic Investigations of WhatsApp Web Utilizing IndexedDB Persistent Storage," *Future Internet, vol. 12, no. 11, p. 184,* https://www.semanticscholar.org/paper/Browser-Forensic-Investigations-of-WhatsApp-Web-Paligu-Varol/0054508526255eff5c15de5ab3194591e842d731, 2020, Last accessed September 18th 2026

[13] Oleg Afonin, "Web Browser Forensics in Digital Triage," *ElcomSoft Blog,* https://blog.elcomsoft.com/2026/01/web-browser-forensics-in-digital-triage, January 2026, Last accessed September 18th 2026

[14] Ilya Kobzar, "LevelDB WAL log: extracting ChatGPT desktop conversations," *Ilya Kobzar Blog,* https://www.ilyakobzar.com/p/leveldb-wal-log-extracting-chatgpt, 2024, Last accessed September 18th 2026

[15] F. Paligu and C. Varol, "Browser Forensic Investigations of Instagram Utilizing IndexedDB Persistent Storage," *Future Internet, vol. 14, no. 6, p. 188,* https://www.mdpi.com/1999-5903/14/6/188, June 2022, Last accessed September 18th 2026

[16] Magnet Forensics, "UserAssist Forensic Artifacts: What They Are and How to Use Them," *Magnet Forensics Blog,* https://www.magnetforensics.com/blog/artifact-profile-userassist, 2023, Last accessed September 18th 2026

[17] Seth Enoka, "ShellBags Forensics: Windows User Navigation Evidence," *Seth Enoka Digital Forensics,* https://sethenoka.com/shellbags-and-user-navigation-what-windows-remembers-about-exploration, 2023, Last accessed September 18th 2026

[18] Nicole Ibrahim, "Available Artifacts, Evidence of Execution," *1234n6 Blog,* https://blog.1234n6.com/available-artifacts-evidence-of-execution, 2023, Last accessed September 18th 2026

[19] Forensafe, "UserAssist Forensic Analysis," *Forensafe Blog,* https://forensafe.com/blogs/UserAssist.html, 2023, Last accessed September 18th 2026

[20] M. W. Kankanamge, N. McKenna, S. Challa, and A. Imteaj, "Digital Forensic Investigation of the ChatGPT Windows Application (ResearchGate Repository)," *ResearchGate,* https://www.researchgate.net/publication/392315098_Digital_Forensic_Investigation_of_the_ChatGPT_Windows_Application, May 2025, Last accessed September 18th 2026

[21] OpenAI, "ChatGPT Documentation: Computer History," *OpenAI Help Center,* https://learn.chatgpt.com/docs/customization/computer-history, 2025, Last accessed September 18th 2026

[22] Oxygen Forensics, "How to Extract and Parse Data from ChatGPT with Oxygen Forensics," *Oxygen Forensics Technical Resources,* https://www.oxygenforensics.com/technical-resources/how-to-extract-and-parse-chatgpt-data, 2024, Last accessed September 18th 2026

[24] Forensics Wiki, "Mac OS X 10.9 Artifacts Location," *Forensics Wiki,* https://forensics.wiki/mac_os_x_10.9_artifacts_location, 2022, Last accessed September 18th 2026

[25] Elite Digital Forensics, "Mac Forensic Analysis Explained," *Elite Digital Forensics,* https://elitedigitalforensics.com/mac-forensic-analysis-explained, 2024, Last accessed September 18th 2026

[26] Arsenal Recon, "Quick Look Cache Parsing," *Arsenal Recon Insights,* https://arsenalrecon.com/insights/quick-look-cache-parsing, 2023, Last accessed September 18th 2026

[27] Patrick Wardle, "'quicklook' - Cache May Leak Encrypted Data," *Objective-See Blog,* https://objective-see.org/blog/blog_0x30.html, June 2018, Last accessed September 18th 2026

[28] Sara Newcomer, "QuickLook Thumbnail Cache Database Forensic Analysis," *Issues in Information Systems (IACIS), vol. 15, no. 2, pp. 421–430,* https://iacis.org/iis/2014/10_iis_2014_421-430.pdf, 2014, Last accessed September 18th 2026

[29] Oleg Afonin, "The C:\User Data in Windows Forensics," *ElcomSoft Blog,* https://blog.elcomsoft.com/2026/03/the-cuser-data-in-windows-forensics, March 2026, Last accessed September 18th 2026

[30] Malwarebytes, "Researcher Claims Claude Desktop Installs 'Spyware' on macOS," *Malwarebytes Labs,* https://www.malwarebytes.com/blog/news/2026/04/researcher-claims-claude-desktop-installs-spyware-on-macos, April 2026, Last accessed September 18th 2026

[31] Anthropic, "Install Claude Desktop," *Anthropic Help Center,* https://support.claude.com/en/articles/10065433-install-claude-desktop, 2025, Last accessed September 18th 2026

[32] Psmths, "windows-forensic-artifacts: Windows Forensic Artifacts Reference Catalog," *GitHub Repository,* https://github.com/Psmths/windows-forensic-artifacts, 2025, Last accessed September 18th 2026

[33] Bhupinder Singh, "A Forensic Analysis of Volatility and Persistence in Windows 11 Copilot Artifacts," *ResearchGate Preprint,* https://www.researchgate.net/publication/399256167_A_Forensic_Analysis_of_Volatility_and_Persistence_in_Windows_11_Copilot_Artifacts, December 2025, Last accessed September 18th 2026

[34] Timothy Kang, "Apps Leave Traces: Forensic Insights from the Microsoft Store," *Detect FYI,* https://detect.fyi/apps-leave-traces-forensic-insights-from-the-microsoft-store-b771c054c14c, February 10th, 2026, Last accessed September 18th 2026

[35] LevelBlue SpiderLabs, "Windows Search Index: The Forensic Artifact You've Been Searching For," *LevelBlue SpiderLabs Blog,* https://www.levelblue.com/blogs/spiderlabs-blog/windows-search-index-the-forensic-artifact-youve-been-searching-for, 2024, Last accessed September 18th 2026

[36] Howard Chivers and Christopher Hargreaves, "Forensic data recovery from the Windows Search Database," *Digital Investigation, vol. 8, no. 1, pp. 44–52,* https://www.sciencedirect.com/science/article/abs/pii/S1742287611000028, 2011, Last accessed September 18th 2026

[37] Digital Detective, "Forensic Analysis of the Zone.Identifier Stream," *Digital Detective,* https://www.digital-detective.net/forensic-analysis-of-zone-identifier-stream, 2020, Last accessed September 18th 2026

[38] Microsoft Corporation, "[MS-FSCC]: Zone.Identifier Stream Name," *Microsoft Learn Technical Documentation,* https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/6e3f7352-d11c-4d76-8c39-2516a9df36e8, 2024, Last accessed September 18th 2026

[39] Shariq Murtuza, "Forensic Implications of Localized AI: Artifact Analysis of Ollama, LM Studio, and llama.cpp," *arXiv preprint arXiv:2603.23996,* https://arxiv.org/html/2603.23996v1, March 25th, 2026, Last accessed September 18th 2026

[40] Sungjo Jeong, Sangjin Lee, and Jungheum Park, "LangurTrace: Forensic analysis of local LLM applications," *Forensic Science International: Digital Investigation, vol. 55, p. 301889,*

https://www.sciencedirect.com/science/article/pii/S2666281725001271, October 2025, Last accessed September 18th 2026

[41] Raymond Peck, "Unlocking Your Claude History, Part 1," *Medium,* https://medium.com/@raymondpeck/unlocking-your-claude-history-part-1-f19000c05655, 2024, Last accessed September 18th 2026

[44] Oleg Afonin, "Investigating Windows Registry," *ElcomSoft Blog,* https://blog.elcomsoft.com/2026/02/investigating-windows-registry, February 2026, Last accessed September 18th 2026

[48] A. Satter, M. Salmon, L. Muhanna, T. Spinosa, T. Gharaibeh, and I. Baggili, "With or without logs: Memory Forensic Reconstruction of Model Context Protocol (MCP) Activity in Agentic LLM Systems," *Forensic Science International: Digital Investigation, vol. 56, p. 301931,* https://www.sciencedirect.com/science/article/pii/S2666281726000879, 2026, Last accessed September 18th 2026