# Phase 1 — Technical Research Notebook

**Project:** *The Algorithmic Panopticon: How AI Amplifies Security Exploits in Online Communities* **Phase:** 1 — Technical & Systems Analysis **Owner / Team:** Mohammad Wael (and team) **Date Created:** 2025-10-18

**How to use this notebook:** - Treat this as your living lab notebook for Phase 1. Keep entries chronological and annotate every external source with URL + short note.  
- **DO NOT** include real users’ PII or run third-party plugins on real accounts. Work only with public code and controlled dummy accounts.

## Table of Contents

1. Scope & Research Questions
2. Annotated Bibliography (public sources)
3. Plugin Audit Table (static analysis)
4. Cache & Persistence Observations (sandbox)
5. AI Exploitation Model Templates
6. Findings Synthesis (2–3 page summary)
7. IRB Notes & Ethics Checklist
8. To‑Do / Action Items

## 1. Scope & Research Questions

**Scope (short):** Discord (primary), public client-side mods (e.g., Vencord), ephemeral messaging behaviors, local cache artifacts, and conceptual AI pipelines.

**Primary Research Questions:** - Which client-side plugin features explicitly or implicitly bypass platform security/privacy controls?  
- What artifacts of “deleted” or “ephemeral” messaging persist on local systems or in transit?  
- What data formats, identifiers, or metadata enable automated ingestion/correlation by an AI?  
- What constraints limit large-scale automated exploitation today?  
- What realistic AI pipelines (inputs → processing → outputs) could scale these vulnerabilities?

## 2. Annotated Bibliography (public sources)

Add at least 8–10 entries here. For each entry use this mini-template:

* **Title / Source:**  
  **URL:**  
  **Type:** (e.g., GitHub repo, DFIR blog, Discord API docs, Reddit thread)  
  **Short note (3 bullets):**

## 3. Plugin Audit Table (static analysis)

*Instructions:* Only perform **static** reviews of public repo code and docs. Record commit hashes or release tags when possible.

| **Plugin** | **Version / Commit** | **Feature** | **Hook(s) Observed** | **Stores Locally?** | **Data Types** | **Security Control Bypassed** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MessageLogger (for Vencord) | “Temporarily logs deleted and edited messages.” ([vencord.dev](https://vencord.dev/plugins/MessageLogger?utm_source=chatgpt.com" \o "MessageLogger - Vencord)) | Logs deleted & edited messages; allows searching message history in mods. ([GitHub](https://github.com/Syncxv/vc-message-logger-enhanced?utm_source=chatgpt.com)) | Hooks into message‐delete / message‐edit events (e.g., onMessageDelete) – see commit “messageLogger: fix ignore guild (#1632)” ([git.derg.cz](https://git.derg.cz/ulysia/Vencord/commit/6e7996659f2eeab8062cccf82ee7af6df1c69497?utm_source=chatgpt.com)) | Yes — plugin description says it saves logs to file or IndexedDB. ([GitHub](https://github.com/Syncxv/vc-message-logger-enhanced?utm_source=chatgpt.com)) | Text (deleted message content), message IDs, timestamps, channel IDs, user IDs, attachments. ([GitHub](https://github.com/Syncxv/vc-message-logger-enhanced?utm_source=chatgpt.com)) | Expected deletion of messages won’t remove them from the client’s log history | The plugin is open source; commit 6e7996… shows code changes in src/plugins/messageLogger/index.tsx ([git.derg.cz](https://git.derg.cz/ulysia/Vencord/commit/6e7996659f2eeab8062cccf82ee7af6df1c69497?utm_source=chatgpt.com)) |
| Discord Desktop Client Cache | — (version unspecified) | Persisting cached content after message/file deletion in official client. ([Pen Test Partners](https://www.pentestpartners.com/security-blog/discord-as-a-c2-and-the-cached-evidence-left-behind/?utm_source=chatgpt.com)) | Not a plugin hook per se – the official client’s cache system stores attachments and message data in a Chromium “Simple Cache” format. ([Pen Test Partners](https://www.pentestpartners.com/security-blog/discord-as-a-c2-and-the-cached-evidence-left-behind/?utm_source=chatgpt.com)) | Yes — as documented: %AppData%\Discord\Cache\Cache\_Data and data\_# / f\_\* files remain. ([Pen Test Partners](https://www.pentestpartners.com/security-blog/discord-as-a-c2-and-the-cached-evidence-left-behind/?utm_source=chatgpt.com)) | Cached attachments, image thumbnails, message fragments, webhook URLs, API call residues. ([Pen Test Partners](https://www.pentestpartners.com/security-blog/discord-as-a-c2-and-the-cached-evidence-left-behind/?utm_source=chatgpt.com)) | Deletion of a message in the UI does *not* guarantee removal from local cache. ([LJMU Research Online](https://researchonline.ljmu.ac.uk/id/eprint/13845/3/Digital%20Forensic%20Acquisition%20and%20Analysis%20of%20Discord%20Applications.pdf?utm_source=chatgpt.com)) | This is outside of a mod – this demonstrates a platform‐native persistence vulnerability |
| **AlwaysTrust** | (Plugin as listed on Vencord Plugins page) ([vencord.dev](https://vencord.dev/plugins/AlwaysTrust?utm_source=chatgpt.com" \o "AlwaysTrust - Vencord)) | Removes the “untrusted domain” & “suspicious file” pop-up warnings in the client. | Not explicitly documented which event hooks, but the code path shows patching of “suspicious file popup” methods. ([git.catvibers.me](https://git.catvibers.me/wing/Bencord/src/tag/v1.2.1/src/plugins/alwaysTrust.ts?utm_source=chatgpt.com)) | N/A (not clearly stated) | Domain links, file trust prompts | User‐interface / security warning bypass | Exists to override warnings about unsafe files/links. |
| **AnonymiseFileNames** | Commit b3819228ed… in Vencord repo (Mar 2024) ([git.derg.cz](https://git.derg.cz/ulysia/Vencord/src/commit/b3819228eddd276ae88d14faa63aa5304d6ac518/src/plugins/anonymiseFileNames/index.tsx?utm_source=chatgpt.com)) | Anonymises uploaded file names (randomises or replaces them). | Upload file path modifications (uploadFiles:(...args)) in plugin code. ([git.derg.cz](https://git.derg.cz/ulysia/Vencord/src/commit/b3819228eddd276ae88d14faa63aa5304d6ac518/src/plugins/anonymiseFileNames/index.tsx?utm_source=chatgpt.com)) | N (it changes file name before upload) | Filenames, attachment metadata | Integrity/traceability of uploaded files; reduces forensic trace of source filename | Script modifies upload process; shows how metadata can be changed client-side. |
| **MessageLogger** | Public Vencord plugin by rushii, V, AutumnVN, etc. ([vencord.dev](https://vencord.dev/plugins/MessageLogger?utm_source=chatgpt.com" \o "MessageLogger)) | Logs deleted and edited messages (“temporarily logs deleted and edited messages”). | Hooks onMessageDelete, onMessageEdit as indicated in community wiki. ([wiki.vencord.dev](https://wiki.vencord.dev/index.php?mobileaction=toggle_view_desktop&title=MessageLogger&utm_source=chatgpt.com" \o "MessageLogger - Discord Client Modding Wiki)) | Y — saves messages to JSON/IndexedDB according to GitHub repo. ([GitHub](https://github.com/Syncxv/vc-message-logger-enhanced?utm_source=chatgpt.com)) | Text content, message IDs, timestamps, attachments | The UI expectation that deleted message = gone is bypassed | Very clear case of client-side logging of deleted data for later retrieval. |

**Audit checklist (for each plugin)** - Repo path(s) inspected:  
- Manifest / metadata (author, license, version):  
- Event hooks & API usage (list):  
- Local storage methods (filesystem, DB, IndexedDB):  
- Network calls (endpoints, hosts) — *only if visible in code*:  
- Any obfuscation / minified bundles:

| **Plugin** | **Code path / hook detail** | **Storage / persistence mechanism** |
| --- | --- | --- |
| **vc‑message‑logger‑enhanced** | In the GitHub repo *Syncxv/vc-message-logger-enhanced*, files such as LoggedMessageManager.ts and index.tsx handle deletion hooks. ([GitHub](https://github.com/Syncxv/vc-message-logger-enhanced?utm_source=chatgpt.com)) The index.tsx sets up event listeners for message deletion and editing (e.g., onMessageDelete, onMessageEdit). | The README states that the plugin “saves messages to a json file” (in earlier versions) and later “moved from JSON to IndexedDB” in version 4.0.0. ([GitHub](https://github.com/Syncxv/vc-message-logger-enhanced?utm_source=chatgpt.com)) So local storage = IndexedDB (or JSON file fallback). |
| **AlwaysTrust** | In the Vencord plugin path src/plugins/alwaysTrust.ts (version v1.2.1) the code patches client methods: e.g., it finds .displayName="MaskedLinkStore" then replaces .isTrustedDomain=function(){return true} effectively overriding the domain‐trust check. ([Catvibers Git](https://git.catvibers.me/wing/Bencord/src/tag/v1.2.1/src/plugins/alwaysTrust.ts?utm_source=chatgpt.com" \o "Bencord/src/plugins/alwaysTrust.ts at v1.2.1 - Catvibers Git)) | No explicit data‐logging storage; the plugin changes UI/logic at runtime — i.e., bypasses alerts/trust checks rather than storing user content. |
| **AnonymiseFileNames** | Code is in src/plugins/anonymiseFileNames/index.tsx within the Vencord repo; it hooks into the upload path (uploadFiles handler) and transforms filenames before they’re sent. (commit b3819228…) | Doesn’t store files locally — transforms metadata client‐side pre‐upload. So the storage mechanism is “network/metadata modification” rather than local store. |
| **MessageLogger** (the simpler/plugin listing) | On the Vencord Plugin page: “Temporarily logs deleted and edited messages.” ([vencord.dev](https://vencord.dev/plugins/MessageLogger?utm_source=chatgpt.com" \o "MessageLogger)) The code path corresponds to the plugin’s event hooks for onMessageDelete and onMessageEdit. | Similar storage behavior as enhanced version: presumably local logs (JSON/IndexedDB) of deleted/edited messages. |
| **Discord Desktop Client Cache Artifact** | For the official client: cached files under %AppData%\Discord\Cache\Cache\_Data, data\_\*, f\_\*. DFIR blog posts show these contain message fragments/attachments. ([Reddit](https://www.reddit.com/r/discordapp/comments/15cwzzi/is_there_anyway_to_stop_somebody_from_seeing/?utm_source=chatgpt.com)) | Local storage = client cache (browser/Chromium‐style). Persisted after deletion unless manually cleared; not tied to a plugin. |

## 4. Cache & Persistence Observations (sandbox)

*Instructions:* Use a private test server and dummy accounts. Do **not** enable or run third-party plugin binaries against other people’s systems. Capture OS paths, filenames, timestamps.

| Artifact Path | OS | Observed Content | Persisted After Deletion? (Y/N) | Recovery Method (public ref) | Notes |
| --- | --- | --- | --- | --- | --- |
| e.g., %AppData%/Discord/Cache/xxx | Windows | Partial message fragments | Y | DFIR blog post 2022 | timestamps retain message index |

**Sandbox checklist** - OS & client version used:  
- Dummy account IDs (anonymized labels):  
- Steps performed (send msg → delete → inspect cache):  
- Tools used (File Explorer, Wireshark passive capture, etc.):  
- Screenshot references (stored locally) — label each screenshot:

## 5. AI Exploitation Model Templates

For each conceptual model, complete the template below and draw a simple ASCII or Mermaid diagram if helpful.

### Model: Recon Agent (example)

**Inputs:** deleted message logs (text), message metadata (timestamps, msgIDs), user metadata (display names) — *all anonymized in research outputs.*

**Processing Steps (high level):** 1. Preprocessing: normalize text, remove stopwords, mask obvious PII. 2. Embedding & clustering: semantic embeddings → cluster by topic/personality indicators. 3. Behavioral inference: apply psycholinguistic models to infer traits. 4. Output: persona summaries, high‑value target lists.

**Outputs:** persona profiles (non-PII), risk scores, recommended social-engineering vectors (***do not*** implement or provide step‑by‑step attack instructions).

**Feasibility:** High/Medium/Low — justify with references.

**Ethical/Harms:** Describe potential harms and mitigation strategies.

**Mermaid diagram suggestion:**

flowchart LR  
 A[Deleted logs] --> B[Preprocess]  
 B --> C[Embeddings & Clustering]  
 C --> D[Behavioral Inference]  
 D --> E[Persona Profiles]

## 6. Findings Synthesis (2–3 page summary)

Use this section to draft the technical narrative you will hand off to Phase 2. Structure suggestions: - Executive summary (1 paragraph) - Key technical findings (bulleted) - What assumptions users make vs. reality - Potential AI amplification vectors (high-level, non-actionable) - Limitations & open questions

## 7. IRB Notes & Ethics Checklist

* Work restricted to public repos and controlled sandbox.
* No collection of real user data.
* All forum content paraphrased and anonymized.
* All outputs will be non-actionable and high-level.

**IRB draft bullet points:** - Study aims and non-malicious intent.  
- Data sources (public code, sandbox artifacts, paraphrased public forum posts).  
- Privacy & de-identification steps.  
- Risk mitigation and storage policies.

## 8. To‑Do / Action Items (editable checklist)

* Create sandbox server & 3 dummy accounts (anonymized labels).
* Populate annotated bibliography (≥8 sources).
* Fill plugin audit rows for MessageLogger, AlwaysTrust, and 3 other plugins.
* Run sandbox cache test and populate artifact table.
* Draft Recon Agent & Forensic Aggregator one-pagers.
* Draft 2–3 page technical summary for Phase 2.

**Appendix: Quick safe-research reminders** - Never store screenshots or raw logs that include real usernames or real server names in shared drives without anonymization.  
- When paraphrasing forum posts, change wording and remove unique phrases to prevent search-based re-identification.  
- If uncertain, include the question in IRB notes.

*End of notebook template.*