A Forensic Analysis of the WebView2-based Microsoft Teams Client

Chiara Bertolotti[1], Mattia Epifani[1] and Giovanni Lagorio[1,*]

[1]University of Genoa, Genoa, Italy

Abstract

The widespread adoption of Microsoft Teams in corporate, educational, and governmental sectors has established it as a critical source of digital evidence. In 2023, Microsoft transitioned the Teams desktop client from an Electron framework to a WebView2 architecture, fundamentally altering data storage by leveraging the shared Microsoft Edge environment. This shift rendered many existing forensic tools and methodologies obsolete, creating a significant knowledge gap for investigators.

This paper addresses this gap by presenting a systematic forensic analysis of the new WebView2-based Teams client on Windows. Using a black-box methodology in a controlled environment, we identified and mapped the new storage locations and data formats for key user activities. Our findings demonstrate that crucial artifacts, including chats, file transfers, and call metadata, can be reliably recovered from new data structures.

To translate these findings into a practical solution, we developed and validated an open-source workflow and a custom Autopsy plugin for automated artifact extraction and analysis. This work provides the digital forensics community with both an updated map of forensic artifacts for the new Microsoft Teams architecture and an immediately usable tool to ensure investigative capabilities keep pace with evolving application platforms.

Keywords

Microsoft Teams, WebView2, Electron, LevelDB, IndexedDB, Windows

1. Introduction

The rapid evolution of collaboration platforms presents a persistent challenge for digital forensics. Microsoft Teams, widely adopted by organizations worldwide [1], according to Business of Apps, has become a critical source of digital evidence in legal, corporate, and governmental investigations, making the ability to analyze its artifacts essential for modern casework.

However, a significant architectural shift in 2023 rendered many established forensic methodologies for Teams obsolete: Microsoft transitioned the Windows desktop client from the self-contained Electron framework to a WebView2 architecture, which leverages the shared Microsoft Edge browser environment for data storage. This fundamental change invalidated existing tools and created a critical knowledge gap for investigators, as current public and academic research on the forensic implications of this new architecture is limited.

This paper directly addresses this forensic void by using a black-box methodology in a controlled environment. We conducted a systematic analysis of the WebView2-based Teams client to determine if core artifacts like chat messages and call metadata could still be reliably extracted. Our primary objective was to map the new data storage locations and formats. Our findings confirm that crucial evidence can be recovered. To make these findings actionable, we developed an open-source workflow and a custom Autopsy plugin for automated analysis.

The key contributions of this work are:

Updated Artifact Map A comprehensive catalog of new data storage locations and formats for forensically relevant artifacts within the WebView2-based Teams client on Windows.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 9-13, 2026, Cagliari, IT

*Corresponding author.

bertolottichiara@libero.it (C. Bertolotti); mattia.epifani@realitynet.it (M. Epifani); giovanni.lagorio@unige.it (G. Lagorio)

0009-0007-4877-5736 (C. Bertolotti); 0009-0006-5390-0952 (M. Epifani); 0000-0002-6632-1523 (G. Lagorio)

Methodological Insight Detailed analysis of how user data is structured within the Microsoft Edge cache, providing the necessary knowledge for manual and automated data extraction.

Practical Forensic Solution The development of a novel, open-source workflow and a custom Autopsy plugin for the automated parsing and analysis of these new data formats, ensuring investigative tools remain compatible with the evolving platform.

The remainder of this paper is organized as follows. Section 2 reviews existing literature on Teams forensics, highlighting the research gap this study addresses. Section 3 provides the technical background on the architectural shift from Electron to WebView2 and its forensic implications. Section 4 details the systematic methodology used for artifact identification and analysis. Section 5 presents our key findings, including the updated artifact map, while Section 6 describes the architecture of our open-source Autopsy plugin. Finally, Section 7 summarizes our contributions and outlines directions for future research.

2. Related Work

The existing body of research on Microsoft Teams digital forensics has exclusively targeted the now-deprecated Electron-based architecture. Pioneering work by Paligu and Varol [2] established a foundational analysis, demonstrating that critical artifacts like messages and call logs were stored in unencrypted LevelDB databases. Their study confirmed that a comprehensive dataset of user activity, including timestamps and communication metadata, could be reliably extracted even without user authentication.

Building on this, Bowling et al. [3] conducted a comparative, multi-platform forensic analysis of Teams on Windows, Android, and iOS. Their research revealed significant disparities in the available forensic data across operating systems, concluding that the Windows desktop version offered the most complete dataset. This finding underscored the necessity of platform-specific investigative strategies.

Another key contribution came from Alexander Bilz [4], who provided a technical walkthrough that mapped the local artifacts of the Windows desktop client. This work reaffirmed that data, such as chat messages and session tokens, persisted in plaintext within LevelDB-backed stores. To streamline the investigative process for the community, Bilz also developed custom Python scripts and an Autopsy plugin to automate artifact extraction.

While this foundational research is invaluable, its applicability is confined to the legacy Electron architecture.

3. Background

The transition of the Microsoft Teams client from an Electron to a WebView2 architecture represents a pivotal shift for forensic analysis. The former Electron framework is an open-source solution that bundles a full Chromium browser engine and a Node.js runtime environment [5], allowing for the creation of cross-platform applications for Windows, macOS, and Linux. While this approach ensures portability, it results in large application sizes and high memory consumption.

In contrast, WebView2 is a Microsoft component that embeds a Chromium-based web view into a native Windows application. Rather than bundling its own rendering engine, WebView2 leverages the Microsoft Edge browser already installed on the system, which significantly reduces the application's footprint and improves performance, as reported in the Microsoft Teams Blog [6]. This architectural difference is forensically critical: data storage locations and file formats are no longer self-contained within the application but are instead influenced by the shared, system-level browser environment, as summarized in Table 1.

Despite this framework change, forensic analysis remains grounded in the fact that both architectures are built on a Chromium engine. Consequently, digital traces can still be found in the browser's disk cache and Web Storage. This mechanism facilitates local data storage and includes Local Storage, for

Table 1
Comparison of WebView2 and Electron frameworks

Category	WebView2	Electron
Platform support	Windows only (requires Edge/Chromium)	Cross-platform: Windows, macOS, Linux
Bundled runtime	Can reuse system runtime or include a fixed version	Includes Chromium + Node.js, higher resource usage
Architecture	Single host process with integrated rendering	Multi-process architecture (main + renderer) with IPC
OS functionality access	Requires explicit bindings to native APIs	Node.js integration + native modules available
Application size	Small (tens of MB)	Large (hundreds of MB)
Security model	Controlled by the host application	Sandboxing and process isolation
Typical use case	Hybrid native Windows applications (e.g., Teams)	Cross-platform standalone applications (e.g., Slack, VS Code)
Notable users	Teams, Outlook, PowerPoint	Slack, Discord

Figure 1: Microsoft Teams IndexedDB's Databases with Object Stores

data that persists indefinitely, and IndexedDB, which functions as a client-side database for structured data.

IndexedDB is a high-level API standardized by the W3C that organizes data into *object stores*, data containers comparable to tables in relational databases; Figure 1 shows an example of such structures. Unlike traditional tables, however, object stores are schema-independent and can hold diverse objects, making them well-suited for storing complex application data. In Chromium-based browsers, IndexedDB is implemented on top of LevelDB, a fast key-value storage library. Applications like Microsoft Teams use IndexedDB to store chat histories, user preferences, and configuration data, which are then backed by LevelDB files on disk.

The persistence model of LevelDB is particularly relevant for digital forensics. Artifacts may remain in LevelDB files even after a user closes an application or deletes browsing data, as records are not immediately overwritten or securely deleted. Additionally, LevelDB log files may retain obsolete or deleted records. Therefore, a strong understanding of how LevelDB structures and stores its data is critical for forensic analysis and data recovery, as discussed in CCL Solutions Group [7].

In summary, while the underlying application frameworks differ, the data storage logic remains tied to Chromium principles. This study focuses on the challenge of analyzing these principles within the new Teams and Edge context, as the existing literature has become outdated and addresses only the former Electron architecture.

4. Methodology

This section outlines the systematic methodology used to acquire and analyze digital artifacts from the WebView2-based Microsoft Teams client. The process was designed to be rigorous and reproducible, centered on a black-box approach to identify key forensic artifacts and evaluate the effectiveness of current forensic tools.

4.1. Test Environment and Research Question

This research was carried out in a controlled laboratory environment, focusing on the Microsoft Teams desktop client for Windows (version 25072.125072.1611.3570), which was the current version during the initial experimental phase in October 2025. Although the core analysis targeted this specific build, the tool has since been consistently maintained and updated. It was successfully validated against the latest production release as of February 2026, confirming its ongoing applicability.

This study was guided by a primary research question stemming from the knowledge gap created by the architectural transition: *Does the new WebView2-based Microsoft Teams client store forensically significant artifacts that can be used to support timeline reconstruction in forensic investigations?*

4.2. Data Collection and Artifact Identification

To generate a comprehensive set of digital artifacts, a series of controlled user actions were systematically performed within the test environment. These actions simulated typical user behavior, covering one-on-one chats, group chats, and channel communications. Activities included sending and editing messages, sharing files, organizing meetings, and managing user permissions. Rather than executing all actions at once and performing a single extraction at the end, which could have led to partial data loss, tests were deliberately conducted in small batches. After each batch, a forensic image of the system's hard drive was acquired using FTK Imager by Exterro [8]. This iterative acquisition process ensured that artifacts generated at each stage were reliably captured, minimizing the risk of overwriting or missing volatile or transient data.

From the simulated actions, a preliminary behavioral analysis allowed us to define a set of forensically relevant artifacts. These included:

Communication Calls, meetings, and text messages (original, edited, deleted, reacted to, and forwarded), as well as file sharing.

Collaboration Creation of group chats and channels, member management, mentions, and the use of tags.

The complete artifact list is reported in the Appendix A.

The analysis of the acquired data proceeded in two phases. The first phase involved a manual, low-level inspection of Teams' primary storage structures—Local Storage, IndexedDB, and the Disk Cache, to identify data repositories (as illustrated in Figure 2). In the second phase, these manual findings were validated using specialized forensic tools, including a commercial solution and our custom Autopsy plugin, which we describe in detail in Section 6.

4.3. Tool Selection and Critical Evaluation

A critical evaluation of several open-source and commercial tools was conducted to identify the most suitable solutions for this analysis. Existing free tools like *forensicsim* [9] and *LevelDBDumper* [10] proved to be ineffective due to changes in data serialization formats and, a lack of support for the new `idb_cmp1` comparator[1], respectively.

Based on this evaluation, the following tools were selected for their proven effectiveness with the new architecture:

[1]It is a user-defined comparison function used by LevelDB to determine the order of keys in IndexedDB-supported databases.

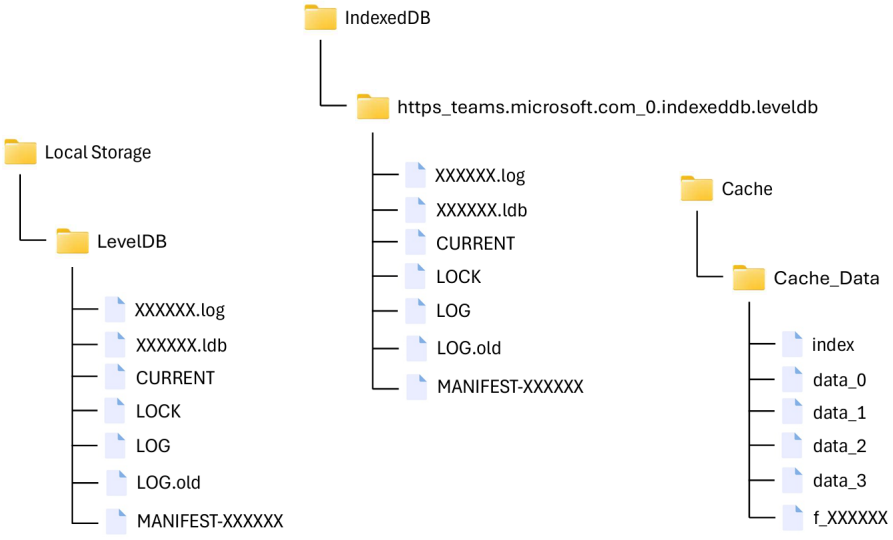

Figure 2: Local Storage, IndexedDB and Cache folders for Microsoft Teams in the WebView2

DFIndexedDB A Python-based tool from Google [11], chosen for its robust ability to parse raw IndexedDB files and produce a structured JSON output. This tool was essential for low-level, in-depth analysis.

Browser History Examiner (BHE) A commercial solution selected for its user-friendly interface, comprehensive multi-artifact extraction capabilities, and its ability to recover deleted data and provide a chronological timeline view. BHE, from Forensics [12], was used for high-level visualization and validation.

While *LevelDB Recon* [13] was also tested and found to be fully functional, BHE was ultimately selected for this study's primary analysis due to its comprehensive visualization and validation capabilities.

5. Results

This section presents the central findings of our forensic investigation, directly addressing the research question outlined in Section 4. The Browser History Examiner was used not only to validate extracted data but also to trace the storage location and internal structure of each artifact, ensuring a transparent and reproducible analytical process.

5.1. Discovery of Forensic Artifacts

Our black-box analysis of the new Teams client identified a comprehensive catalog of digital artifacts, discovering 95 object stores across 56 databases. The key artifacts relevant to forensic timeline reconstruction are summarized in Table 2. These findings provide a crucial, updated resource for investigators, enabling them to locate and extract evidence previously inaccessible with outdated methodologies.

Our investigation focused on three core object stores fundamental to forensic timeline reconstruction and user attribution: *conversations*, *replychains*, and *profiles*. The new architecture stores data in a semi-structured JSON format, which allows for a comprehensive analysis of nested content.

5.2. Conversations

The `conversations` object store provides a high-level view of all active communications and their metadata. Each entry in this store is tied to a unique `conversationId`, which we successfully decoded to classify the communication type. We observed a clear correlation between ID patterns and conversation types:

Table 2
Key Forensic Artifacts and Their Locations

Artifact	Data Location (Object Store)	Forensic Value
Private Chats	`conversations`, `replychains`	Reconstruction of one-on-one communication
Group Chats	`conversations`, `replychains`	Reconstruction of group communication
Channel Activity	`conversations`, `replychains`	Reconstruction of organizational and team-based communication
Other Threads	`conversations`, `replychains`	User-generated content and drafts
Contacts	`profiles`, `conversations`	User profiles and contact information
Voice/Video Calls	`replychains`	Call logs, participants, and event timestamps
File Sharing	`replychains`, `files-thumbnail-blob-conten`	Links to shared files and documents
Reactions/Mentions	`feed-items`, `replychains`	User engagement and social interaction
Pinned Messages	`pinned-messages-store`, `pinnedchannelmessages`, `replychains`	User-defined important messages

- `19:{uid}_{uid}@unq.gbl.spaces` for private chats between two users, the two `uids` are respectively the examined user and the other participant,
- `19:{id}@thread.v2` for group chats,
- `19:{id}@thread.tacv2` for channel conversations,
- `19:meeting_{id}@thread.v2` for meeting chats,
- `48:annotations`,
- `48:calllogs` for video calls in private/group chats,
- `48:drafts` for scheduled messages,
- `48:notes` for messages sent to self,
- `48:notifications` for recent notifications.

This mapping is vital for sorting evidence and establishing a clear context for each message or event. An example of a private conversation is shown in Figure 3.

Key fields include:

- `id`: unique identifier of the thread.
- `properties`: various attributes of the conversation, such as whether the thread is empty or deleted. It also includes the `addedBy` and `addedByTenantId` fields, which indicate the user who added the conversation and their tenant ID.
- `threadProperties`: metadata about the conversation, including whether it is disabled, the topic, the creator, and the creation date. It also contains information about pinned items and tabs associated with the conversation (e.g., integration with other applications), which are useful for summarizing its content.
- `lastMessage`: details of the most recent message in the conversation, including its content, type, timestamp, and sender information. This field can be found in the `replychains` object store as well, allowing for cross-referencing.
- `members`: list of participants along with their IDs and roles.
- `chatTitle`: display name used in the user interface; typically includes the names of participants for private chats, or the group/channel title for group chats and channels.

The described JSON structure is the same for both group conversations and channels, with the only difference being the `type` field, which is set to `Chat` for group conversations and to `Space` or `Topic` for channels and subchannels, respectively. This field can be also set as `Meeting` for meeting chats, `Thread` for `48:annotations`, `48:calllogs` and recent notifications, `Conversation` for scheduled messages and `StreamOfNotes` for messages sent to self.

```
{
  "id": "19:27616d07-acc9-4a75-ad26-312
    ↪ f92a85690_b119c85d-f9f3-4c76-af84-77
    ↪ cf110ad0d6@unq.gbl.spaces",
  "properties": {
    "isDeleted": false,
    "addedBy": "8:orgid:b119c85d-f9f3-4c76-af84
      ↪ -77cf110ad0d6",
    "addedByTenantId": "6cd36f83-1a02-442d-972f
      ↪ -2670cb5e9b1a",
    [...]
  },
  "teamId": "19:27616d07-acc9-4a75-ad26-312
    ↪ f92a85690_b119c85d-f9f3-4c76-af84-77
    ↪ cf110ad0d6@unq.gbl.spaces",
  "threadProperties": {
    "threadType": "chat",
    "isMigrated": true,
    "extensionDefinitionContainer": "{\"
      ↪ updatedTime\":\"1751470175590\"}",
    "tab::6705d385-afa1-4f10-bfd7-976f98557475":
      ↪ "{\"id\":\"tab::6705d385-afa1-4f10-bfd7
      ↪ -976f98557475\",\"order\":110000,\"
      ↪ definitionId\":\"4fa41b04-794c-4e4e-9
      ↪ c06-41618d592f77\",\"directive\":\"
      ↪ extension-tab\",\"name\":\"Kanban%20
      ↪ Task%20Board\",\"type\":\"tab:\",\"
      ↪ settings\":{\"subtype\":\"extension
      ↪ \",\"url\":\"https://kanban-taskboard.
      ↪ com/app?uID=119a89c0a76
      ↪ ...-1751468893925\",\"removeUrl\":\"
      ↪ https://kanban-taskboard.com/removetab?
      ↪ uID=119a89c0a76...-1751468893925\",\"
      ↪ entityId\":\"119a89c0a7\",\"dateAdded
      ↪ \":\"2025-07-02T15:08:16.861Z\"}}",
    "pinnedItems": "[{\"itemId
      ↪ \":\"1753866981322\",\"itemType\":\"
      ↪ Message\"},{\"itemType\":\"Message\",\"
      ↪ itemId\":\"1751469159561\"}]",
    "createdat": "1751468512756",
    "creator": "8:orgid:b119c85d-f9f3-4c76-af84
      ↪ -77cf110ad0d6",
    "tenantid": "6cd36f83-1a02-442d-972f-2670
      ↪ cb5e9b1a",
    [...]
  },
  "lastMessage": {
    "content": "Flameshot",
    "contentHash": "3654886534",
    "isSanitized": true,
    "messagetype": "RichText/Html",
    "contenttype": "Text",
    "activitytype": "",
    "clientmessageid": "1142567388348193084",
    "sequenceId": 72,
    "prioritizeimdisplayname": false,
    "imdisplayname": "Giovanni Lagorio",
    "fromDisplayNameInToken": "Giovanni Lagorio",
    "fromFamilyNameInToken": "Lagorio",
    "fromGivenNameInToken": "Giovanni",
    "properties": {
      "mentions": "[]",
      "cards": "[]",
      "links": "[]",
      "files": "[]",
      "formatVariant": "TEAMS",
      "languageStamp": "languages=en:100;de:93;fr
        ↪ :92;length:9;&detector=Bling"
    },
    "state": 1,
    "id": "1753866981322",
    "type": "Message",
    "composetime": "2025-07-30T09:16:21.322Z",
    "originalarrivaltime": "2025-07-30T09
      ↪ :16:21.322Z",
    "from": "worker/8:orgid:b119c85d-f9f3-4c76-
      ↪ af84-77cf110ad0d6",
    "conversationLink": "conversation/19:27616d07
      ↪ -acc9-4a75-ad26-312f92a85690_b119c85d-
      ↪ f9f3-4c76-af84-77cf110ad0d6@unq.gbl.
      ↪ spaces;messageid=1753866981322",
    "preview": "Flameshot",
    [...]
  },
  "members": [
    {
      "id": "8:orgid:b119c85d-f9f3-4c76-af84-77
        ↪ cf110ad0d6",
      "role": "Admin"
    },
    {
      "id": "8:orgid:27616d07-acc9-4a75-ad26-312
        ↪ f92a85690",
      "role": "Admin"
    }
  ],
  "clientArrivalTime": "2025-08-28T07:35:46.832Z"
    ↪ ,
  "clientUpdateTime": "2025-09-25T08:21:13.128Z",
  "chatTitle": {
    "shortTitle": "Giovanni Lagorio",
    "longTitle": "CHIARA BERTOLOTTI, Giovanni
      ↪ Lagorio",
    "hasUnknownUserInShortTitle": false,
    "avatarUsersInfo": [
      {
        "mri": "8:orgid:b119c85d-f9f3-4c76-af84
          ↪ -77cf110ad0d6",
        "displayName": "Giovanni Lagorio",
        "email": "giovanni.lagorio@unige.it",
        "type": "ADUser",
        "avatarETag": "SignIn_1756366545097"
      }
    ],
    [...]
  },
  [...]
}
```


Figure 3: Example of a private conversation

5.3. Replychains

The `replychains` object store is the most forensically rich artifact we identified, as it contains the actual content of communications. By developing a custom parsing logic for its nested JSON structure, we were able to reconstruct entire conversations.

For instance, a typical message entry within the `replychains` store contains fields such as `content` (for message body), `replyChainId` (a unique ID), and `composeTime` (for timestamps), allowing actions

```
{
  "conversationId": "19:243621021cd243a08e695e40171317fd@thread.v2",
  "replyChainId": "1711374697491",
  "latestDeliveryTime": 1711374697491,
  "parentMessageVersion": 1711374697491,
  "messageMap": {
    "8:orgid:27616d07-acc9-4a75-ad26-312f92a85690_1580948389366606399": {
      "id": "1711374697491",
      "dedupeKey": "8:orgid:27616d07-acc9-4a75-ad26-312f92a85690_1580948389366606399",
      [...]
      "messageType": "RichText/Html",
      "imDisplayName": "CHIARA BERTOLOTTI",
      "creator": "8:orgid:27616d07-acc9-4a75-ad26-312f92a85690",
      "content": "<p>Yes</p>",
      [...]
    }
  }
}
```


Figure 4: Example of a `RichText/Html` message in replychains

to be linked to specific individuals and times. In general, messages can represent various types of content, including activities such as meetings, recordings, file sharing, and member additions. In the extracted dataset, the following event types were identified:

- `Event/Call`
- `RichText/Html`
- `RichText/Media_CallLogRecording`
- `RichText/Media_CallRecording`
- `RichText/Media_CallTranscript`
- `Text`
- `ThreadActivity/AddCustomApp`
- `ThreadActivity/AddMember`
- `ThreadActivity/DeleteMember`
- `ThreadActivity/PinnedItemsUpdate`
- `ThreadActivity/TopicUpdate`

Among these, the most common are `RichText/Html`, `Text` and `Event/Call`, representing text-based messages, direct calls metadata and meeting-related entries, respectively. An example of a simple `RichText/Html` message is shown in Figure 4.

In this example, the message content is a simple HTML paragraph (`<p>Yes</p>`) of type `RichText/Html`, sent by a user identified in the `creator` field. This event type corresponds to a standard text reply in a conversation, but the same schema can be used for other content such as messages with a specific font, or messages containing attachments. Key fields include:

- `content`: message body.
- `imDisplayName`: sender's display name.
- `messageType`: type of message.
- `originalArrivalTime`: message timestamp.
- `creator`: sender's unique ID.
- `isSentByCurrentUser`: whether the message was sent by the analyzed user.

The `messageMap` structure allows handling message threads and replies, especially in channels where posts and comments coexist. Additional fields, such as `properties`, may contain metadata (e.g., detected language, attachments, mentions, edit time of the message, delete time of the message), as shown in Figure 5.

This granularity allows us to reconstruct both the explicit message content and the implicit user actions, such as reactions (`reactions`), mentions (`mentions`) and shared files (`links`), which provide

```
"properties": {
  "mentions": "[]",
  "cards": "[]",
  "importance": "",
  "subject": "",
  "title": "",
  "links": "[{\"@type\":\"http://schema.skype.com/HyperLink\",\"itemid\":\"0\",\"url\":\"https://
      github.com/chiabertolotti/ThesisBertolotti.git\",\"previewenabled\":false,\"preview\":{\"
      previewurl\":null,\"previewmeta\":null,\"description\":null}}]",
  "files": "[]",
  "formatVariant": "TEAMS",
  "languageStamp": "languages=en:100;id:75;fil:73;length:82;&detector=Bling",
  "blurHash": "[{\"imageId\":\"0-frca-d21-06ab34498dd0f2abee3c3e1c3d525cc9\",\"blurhash\":\"L1S?
      DWVy55?JcX~X0KxI9YwN9Y?H\"}]"
}
```


Figure 5: Example of a `properties` field containing link and image in a message

critical context for any investigation. In subsection 5.5 we provide a more detailed overview of these fields.

5.4. Profiles

The `profiles` object store is key for attribution, containing user identity information that links actions from other stores to real-world identities. Each profile includes fields like `displayName` and `email`, which we can use to map a user's unique ID to a real-world identity. An example entry is shown in Figure 6.

For example, by cross-referencing the `creator` field in a message from the `replychains` store with the `displayName` and `email` from the `profiles` store, we can definitively attribute a message to a specific individual. This capability is fundamental for a complete and accurate forensic investigation.

5.5. Other Relevant Artifacts

In addition to the core findings above, our analysis revealed several other significant artifacts that provide valuable contextual information. These discoveries are crucial for painting a complete forensic picture.

Communication Context and Intent

- *Mentions, Reactions, and Direct Replies*: Events related to mentions, reactions, and direct replies are stored in dedicated object stores called `feed-items`. Importantly, these entries contain a `sourceMessageId` that references the original message in `replychains`, facilitating the reconstruction of communication chains and providing a more complete timeline.
- *Pinned Messages*: This artifact is useful for understanding a user's priorities. We discovered that pinned messages are stored in distinct object stores: `pinned-messages-store` for private or group chats and `pinnedchannelmessages` for channels. This provides clear evidence that a user has intentionally flagged a specific message as important.
- *Tags*: Our analysis identified the presence of "tags", which are user-defined aggregations of multiple members to facilitate group mentions.
  These tags are stored in the `people-targeting-tag-cards-store` object store and are crucial for mapping relationships within a specific user group.

```
{
  "alias": "giovanni.lagorio",
  "mail": "giovanni.lagorio@unige.it",
  "objectType": "User",
  "preferredLanguage": "en-US",
  "sipProxyAddress": "giovanni.lagorio@unige.it",
  "smtpAddresses": [
    "Giovanni.Lagorio@cloud.unige.it",
    "Lagorio@dibris.unige.it",
    "Giovanni.Lagorio@dibris.unige.it",
    "Giovanni.Enrico.Lagorio@dibris.unige.it",
    "700231@unige.it",
    "giovanni.lagorio@unige.it"
  ],
  "userPrincipalName": "giovanni.lagorio@unige.it",
  "givenName": "Giovanni",
  "surname": "Lagorio",
  "email": "giovanni.lagorio@unige.it",
  "userType": "Member",
  "tenantName": "unige.it",
  "displayName": "Giovanni Lagorio",
  "mri": "8:orgid:b119c85d-f9f3-4c76-af84-77cf110ad0d6",
  "objectId": "b119c85d-f9f3-4c76-af84-77cf110ad0d6",
  [...]
}
```


Figure 6: Example of a User Profile in Teams

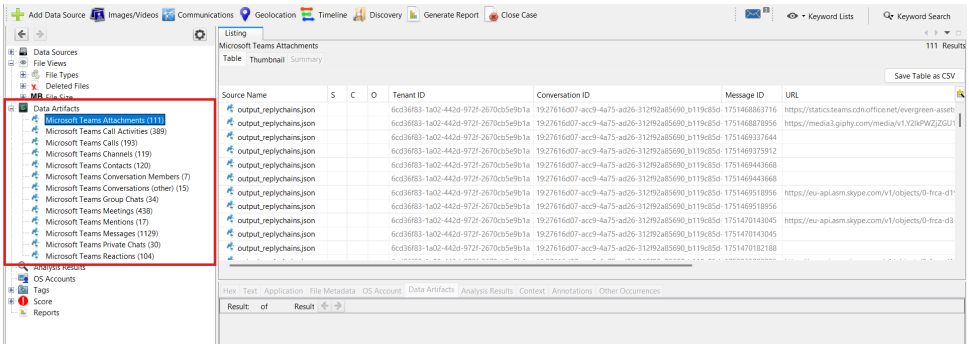

Figure 7: Structured Visualization of Microsoft Teams Artifacts in Autopsy Plugin Output

Activity and Call Metadata

- *Call Logs*: The data for voice and video calls is fragmented but identifiable. It is crucial to distinguish between direct calls and scheduled meetings, as they are managed differently by Teams. Both types generate entries in the `replychains` object store. Direct calls are identified by the ID `48:calllog`, while meetings have a different ID (`19:meeting_id@thread.v2`), allowing investigators to reconstruct a detailed and accurate history of call and meeting activity.
- *Search History*: The `search-history` store contains a record of user searches. While not a direct communication artifact, it can provide valuable clues about an individual's intent or topics of interest and help direct an investigator's focus to specific conversations or files.
- *Favorite contacts*: There is an object store named `capiv3-contacts` that contains a list of contacts added by the user to the "Speed Dial" section of the calling interface.

The complexity of these findings, particularly the correlation between the object stores and their internal subdivision (e.g., messages, call metadata, meetings), necessitated the development of an automated solution, fully described in Section 6.

6. The Autopsy Plugin

To translate our research findings into a practical tool, we developed an open-source workflow and a custom Autopsy plugin for automated analysis of the identified artifacts. As shown in Figure 7, the plugin integrates seamlessly into the Autopsy interface, listing custom artifact categories (e.g., messages, calls, reactions) and allowing investigators to navigate and filter Teams data efficiently. This section details the plugin's motivation, architecture, core functionality, and validation process.

6.1. Motivation and Plugin Architecture

The primary motivation for this development was the absence of a free, open-source tool capable of providing a clear, structured visualization of data from the new Teams architecture. While utilities like DFIndexedDB can extract raw data, they lack an integrated solution for presenting it within a unified forensic environment. Autopsy was selected as the host platform for its modular architecture, active community, and widespread adoption in the digital forensics field.

Our plugin is implemented as a Jython ingest module to balance rapid Python-based development with seamless integration into Autopsy's Java framework. A key architectural decision was to decouple the data processing logic: an independent Python script first uses DFIndexedDB to extract and normalize data from LevelDB files into a structured JSON format. The Autopsy plugin then ingests these pre-processed files, ensuring a clean separation of concerns and future modularity. Detailed instructions for trying our plugin out can be found at: https://github.com/chiabertolotti/teamsartifacts.

To accurately model the heterogeneous nature of Teams data, the plugin dynamically registers custom artifact types and attributes within Autopsy's *Blackboard* system, using a unique `TSK_TEAMS_` prefix. This approach provides several benefits:

- It allows for precise categorization of Teams-specific entities such as messages, calls, reactions, and attachments, which are not covered by Autopsy's standard artifact taxonomy.
- It offers full flexibility to define and store relevant metadata, including conversation IDs, timestamps, and tenant information.
- It prevents conflicts with Autopsy's built-in artifact types, ensuring the integrity of the analysis.

6.2. Core Functionality and Data Processing

The plugin's primary function is to parse the JSON output from the external script and create structured, searchable artifacts within Autopsy. The parsing logic targets the three most forensically significant object stores:

- `conversations`: For conversation metadata and thread mapping.
- `profiles`: For user and contact information.
- `replychains`: For message content, including text, attachments, and reactions.

The plugin processes these stores in a defined sequence to build contextual relationships before creating the final artifacts. It automatically classifies conversation types (e.g., private chat, channel) by analyzing the unique patterns in their conversation IDs.

For each message, the parser applies a multi-level classification and handles various content types:

- Text and HTML Messages: the plugin cleans and normalizes HTML content, extracting plain text while preserving contextual elements like embedded links and quoted replies.
- Call and Meeting Artifacts: messages related to calls and meetings are identified and processed as distinct artifacts, capturing participants, timestamps, and metadata (e.g., call duration or recording status).
- Reactions and Mentions: reaction data (e.g., likes, emojis) and mentions are extracted from message properties, linking them to specific messages and users to enrich the communication timeline.

This robust parsing logic ensures that all relevant information is preserved and made available for investigator analysis.

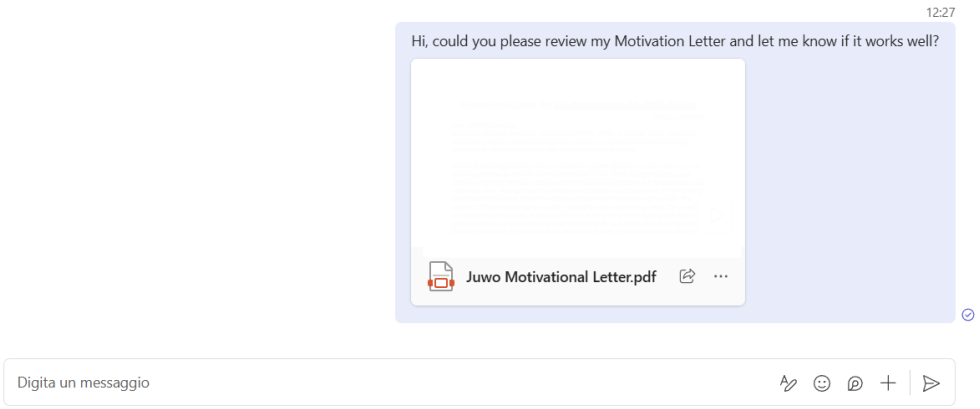

Figure 8: The original artifact (message) as viewed in the Microsoft Teams client

6.3. Testing and Validation

The plugin's reliability was rigorously verified using a dual-testing strategy on two distinct datasets: one controlled and one from a complex, real-world corporate environment. In the controlled dataset, the simulation was carefully designed and documented, meaning that the sequence of user actions, generated messages, and file operations could in principle serve as ground truth. However, to ensure forensic rigor and to detect any discrepancies between expected system behavior and actual artifact persistence, the plugin's output was additionally compared against two industry-standard tools: *Browser History Examiner (BHE)*, previously employed in this investigation and detailed in Section 5, and the commercial suite *Magnet Axiom* [14], which provides a dedicated module for Microsoft Teams artifact analysis. This cross-tool validation ensured that both the simulation was behaving as intended and that the developed plugin aligned with established professional solutions. On the controlled dataset, the plugin demonstrated full consistency with the commercial tools across all key artifacts. This accuracy is visually confirmed by comparing an original Teams message (Figure 8) with its representation in Magnet Axiom (Figure 9) and our plugin's output (Figure 10), which shows that our tool preserves the same level of forensic detail. In the real-world corporate dataset, our plugin successfully retrieved all artifacts recovered by the other tools from the primary tenant, confirming its reliability for large-scale data processing. However, this test also exposed a current limitation: artifacts from secondary tenants were only partially extracted. Overall, the dual-testing approach confirmed that the plugin provides highly accurate and reliable results, while also highlighting the complexity of multi-tenant environments as a primary focus for future development.

7. Conclusions and further work

This paper presents an in-depth forensic analysis of the WebView2-based Microsoft Teams client. Our research demonstrates that the new client stores forensically significant artifacts comparable in value to those of the former Electron-based version, confirming their suitability for investigative timeline reconstruction and validating our methodology.

Our main contributions include the detailed mapping and analysis of 95 identified IndexedDB object stores, providing a comprehensive understanding of their structure and forensic significance. Particular attention was devoted to key object stores such as `replychains`, `conversations`, and `profiles`, as well as previously undocumented data related to features such as pinned messages, search history, and tags. Furthermore, the analysis underscores the current limitations of open-source forensic tools in handling the structural changes introduced by the new architecture.

To bridge this gap, we designed, developed, and validated a custom Autopsy plugin, offering the open-source community a reliable, automated solution for artifact extraction and visualization. The current implementation focuses on the three most forensically relevant object stores, `replychains`,

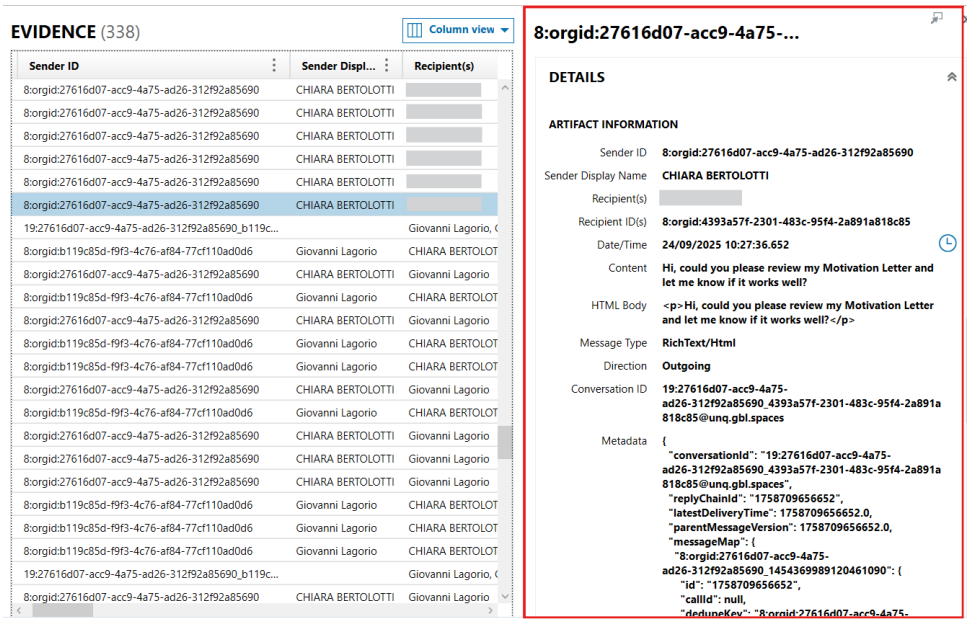

Figure 9: Message shown in Figure 8 extracted through Magnet Axiom (Commercial Tool)

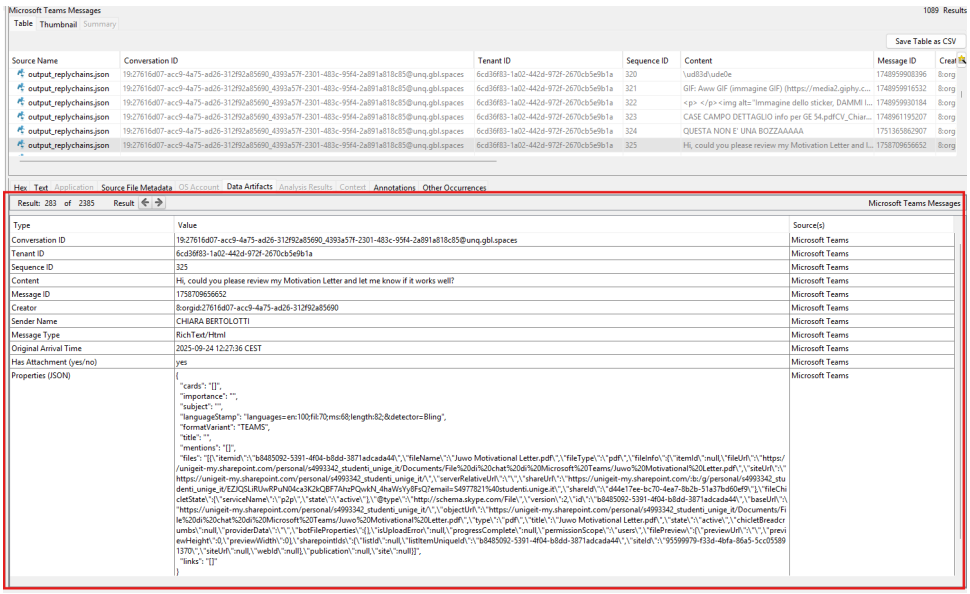

Figure 10: Message shown in Figure 8 extracted through the Autopsy plugin

`conversations`, and `profiles`, which contain the majority of evidentiary data required for timeline reconstruction. Nevertheless, the exhaustive analysis of all 95 stores conducted in this study lays the groundwork for extending the plugin to support additional data structures in future iterations.

While this study establishes a robust framework, several avenues for future work remain. Further research is required to fully clarify the purpose and forensic relevance of all identified structures, including object stores such as `48:annotations`, and to evaluate the recovery potential of draft messages and other transient artifacts. Moreover, as this analysis focused exclusively on Windows, future investigations should extend to other platforms, including the Chromium-based macOS client, the PWA Linux client, and mobile versions for Android and iOS, to establish a comprehensive cross-platform forensic model.

Future development of the Autopsy plugin will focus on expanding artifact coverage and enhancing its performance in complex environments involving multiple accounts and tenants. Additionally, the

plugin could be extended to include Chromium Disk Cache analysis, enabling the recovery of multimedia content and other relevant forensic artifacts.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Grammar and spelling check. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] Business of Apps, Microsoft Teams Revenue and Usage Statistics (2025), https://www.businessofapps.com/data/microsoft-teams-statistics, 2025.

[2] F. Paligu, C. Varol, Microsoft Teams desktop application forensic investigations utilizing IndexedDB storage, Journal of Forensic Sciences (2022).

[3] H. Bowling, K. Seigfried-Spellar, U. Karabiyik, M. Rogers, We are meeting on Microsoft Teams: Forensic analysis in Windows, Android, and iOS operating systems, Journal of Forensic Sciences (2023).

[4] Alexander Bilz, A Forensic Gold Mine III: Forensic Analysis of the Microsoft Teams Desktop Client, 2021. URL: https://www.alexbilz.com/post/2021-09-09-forensic-artifacts-microsoft-teams/, Accessed on March 2025.

[5] Electron Team, Build cross-platform desktop apps with JavaScript, HTML, and CSS, 2025. URL: https://www.electronjs.org/, Accessed on June 2025.

[6] Microsoft Teams Blog, Announcing General Availability of the New Microsoft Teams App for Windows and Mac, 2023. URL: https://techcommunity.microsoft.com/blog/microsoftteamsblog/announcing-general-availability-of-the-new-microsoft-teams-app-for-windows-and-m/3934603.

[7] CCL Solutions Group, Hang on, that's not SQLite: Chrome, Electron and LevelDB, 2020. URL: https://www.cclsolutionsgroup.com/post/hang-on-thats-not-sqlite-chrome-electron-and-leveldb, Accessed on June 2025.

[8] Exterro, FTK Imager, https://www.exterro.com/digital-forensics-software/ftk-imager, 2025.

[9] A. Bilz, forensicsim, https://github.com/lxndrblz/forensicsim, 2021. GitHub repository.

[10] M. Dawson, LevelDBDumper, https://github.com/mdawsonuk/LevelDBDumper, 2024. GitHub repository, GPL-3.0 license.

[11] Google, dfindexeddb, https://github.com/google/dfindexeddb, 2023. GitHub repository, Apache-2.0 license.

[12] F. Forensics, Browser History Examiner, https://www.foxtonforensics.com/browser-history-examiner/, 2025. Version 1.22.1, version history visible online.

[13] Arsenal Recon, Arsenal Recon, https://www.arsenalrecon.com/, 2025. Digital forensics tools including Arsenal Image Mounter, Hibernation Recon, Registry Recon.

[14] Magnet Forensics, Magnet AXIOM, https://www.magnetforensics.com/products/magnet-axiom/, 2025. Digital forensics investigation platform.

A. Appendix

Artifacts
Outgoing/Incoming Calls Answered
Outgoing/Incoming Calls Unanswered
Outgoing/Incoming Calls Recording
Outgoing/Incoming Meetings Answered
Outgoing/Incoming Meetings Unanswered
Outgoing/Incoming Meetings Recording
Creation of Group Chat
Creation of Channel
Add/Remove Members from Group Chat
Add/Remove Members from Channel
Outgoing/Incoming Text Messages Unmodified
Outgoing/Incoming Text Message Reaction
Outgoing/Incoming Text Message Edited
Outgoing/Incoming Text Message Deleted
Outgoing/Incoming Text Message Forwarded
Outgoing/Incoming Text Message Specific Reply
Outgoing/Incoming Single Mention
Outgoing/Incoming Text Messages Non-ASCII Unmodified
Outgoing/Incoming Text Messages Non-ASCII Reaction
Outgoing/Incoming Text Messages Non-ASCII Edited
Outgoing/Incoming Text Messages Non-ASCII Deleted
Outgoing/Incoming Text Messages Non-ASCII Forwarded
Outgoing/Incoming Media File Unmodified
Outgoing/Incoming Media File Reaction
Outgoing/Incoming Media File Deleted
Outgoing/Incoming Media File Forwarded
Outgoing/Incoming Post Unmodified
Outgoing/Incoming Post Reaction
Outgoing/Incoming Post Edited
Outgoing/Incoming Post Deleted
Outgoing/Incoming Post Forwarded
Outgoing/Incoming Comments Unmodified
Outgoing/Incoming Comments Reaction
Outgoing/Incoming Comments Edited
Outgoing/Incoming Comments Deleted
Outgoing/Incoming Comments Forwarded
Outgoing/Incoming Tag Mention