RDPTimeline: Timeline-Based Session Reconstruction and Forensic Analysis Using Windows Event Logs

1. Introduction

Remote Desktop Protocol (RDP) is widely used for remote system administration in Windows environments and is a common component of modern corporate and enterprise world. That also makes it one of the targeted areas in security incidents, including unauthorized access, credential misuse, and post-compromise execution of viruses. As a result, RDP-related artifacts are routinely examined during digital forensic and incident response (DFIR) investigations.

Windows systems generate extensive event logs that record authentication attempts, session activity, account changes, and system modifications. These logs provide valuable forensic evidence, but they are scattered in multiple places and generated by sources that operate independently. Investigators are therefore often required to manually reconstruct timelines and infer and examine relationships between events that were not designed to be interpreted together. While this approach can reveal individual indicators of compromise, it often fails to clearly link system actions to specific remote access sessions. Not to mention it also increases the time required to actually understand the incident.

Most existing RDP forensic approaches focus on authentication events or on ordering events chronologically across log sources. These methods help confirm that remote access occurred but provide limited insight into what happened during a session. Actions such as user creation, privilege escalation, or persistence may appear close in time to an RDP login, but without explicit session boundaries, their attribution remains uncertain, particularly when events are logged with delays or partial context.

This project adopts a session centred approach to RDP forensics. Rather than treating log entries as isolated records, RDP interactions are reconstructed as discrete session objects based on Windows session lifecycle events. System activity occurring within and around these sessions is then connected and correlated using explicit temporal rules. This approach reduces ambiguity caused by asynchronous logging behaviour and helps in presenting a clearer picture of the timeline of incident. The approach also tackles the issue of events logged with delays by introducing a grace time around which incident logs are gathered.

2. Literature Review

Prior research in the domain of digital forensics acknowledges that Windows Event Logs are one of the key sources of timestamped evidence for understanding system and user actions during or after an incident. For example, there are practical guidelines existing for identifying RDP related Event IDs and interpreting their forensic significance, helping investigators track connection, authentication, and session events across multiple logs [1]. Contemporary studies emphasize the importance of Windows logs for reconstructing the sequence of RDP activities, particularly for tracing unauthorized logons, failed authentication events, suspicious activities, and other indicators that support incident response investigations [2].

Additionally, recent work has begun to revisit timeline-based event reconstruction more broadly, addressing existing gaps in terminology and methodology while highlighting key challenges and future research directions in the field [3].

However, recent research still largely treats events independently without focusing on context and gathering information from multiple subsystems. The research still lacks tools that explicitly takes logs during RDP sessions, infers it and correlates multiple log sources into structured session objects and use those objects to analyse the information based on the context**.** This results in increasing the time required to build context of overall incident and reconstruct the timeline of security breach.

The session centric perspective of this project enables coherent forensic interpretation of RDP activity, extending beyond the capabilities supported by existing tools and prior literature.

3. Methodology and System Design

This section presents the methodology and internal design of the proposed DFIR framework. Unlike generic log analysis approaches, the framework is explicitly designed for RDP session reconstruction, with careful selection of logs, events, session management, and forensic rules grounded in Windows session semantics.

3.1. System Architecture Overview

The framework follows a staged pipeline architecture:

1. Log ingestion and validation
2. Event parsing and normalization
3. Global timeline construction
4. RDP session reconstruction
5. Temporal correlation using grace windows
6. Rule based DFIR analysis
7. Optional analytical augmentation (ML and AI reporting)

Each stage consumes the output of the previous stage without introducing hidden state or side effects.

3.2. Required and Optional Log Sources

The framework ingests only RDP relevant Windows Event Logs, deliberately excluding unrelated sources to reduce noise. Log sources are classified as mandatory or optional, depending on their role in session reconstruction.

3.2.1 Mandatory Logs (Core Functionality)

Log Source	**Purpose**	**Justification**
Security.evtx	Authentication, account changes, privilege escalation	Required for identifying login attempts, user manipulation, privilege escalation, and anti-forensics

LocalSessionManager.evtx	Session lifecycle events	Authoritative source for RDP session start and termination

Without these logs, accurate RDP session reconstruction is not possible.

3.2.2 Optional but Strongly Recommended Logs

Log Source	**Purpose**	**Added Value**
RemoteConnectionManager.evtx	RDP authentication and connection context	Corroborates session creation and remote access
System.evtx	Service installation events	Detects service persistence
TaskScheduler.evtx	Scheduled task creation	Detects task persistence

These logs enhance forensic depth but are not strictly required for baseline session reconstruction.

3.3. Event Selection and Extraction Strategy

Rather than ingesting all events, the framework extracts only relevant Event IDs directly related to RDP activity. This targeted extraction improves performance and interpretability.

3.3.1 Authentication and Session Context Events

Event ID	**Log Source**	**Description**	**DFIR Significance**
4624	Security	Successful logon	Contextual (not session start)
4625	Security	Failed logon	Brute-force detection
4634	Security	Logoff	Session termination
1149	RemoteConnectionManager	RDP authentication success	Supporting context
21	LocalSessionManager	Session created	Authoritative session start
24	LocalSessionManager	Session disconnected	Session termination

3.3.2 Account Manipulation Events

Event ID	Log Source	Description	DFIR Significance
4720	Security	User account created	Unauthorized account creation
4722	Security	Account enabled	Account reactivation
4723	Security	Password change attempt	Credential manipulation
4724	Security	Password reset attempt	Privilege abuse
4725	Security	Account disabled	Account tampering
4728	Security	User added to security group	Privilege escalation
4732	Security	User added to administrators	Critical escalation

3.3.3 Persistence and Anti-Forensic Events

Event ID	Log Source	Description	DFIR Significance
4698	Security / TaskScheduler	Scheduled task created	Persistence mechanism
129	TaskScheduler	Task registered	Persistence confirmed
7045	System	Service installed	Persistence via service
1102	Security	Security log cleared	Anti forensic activity

3.4. Event Parsing and Normalization

Each extracted event is converted into a normalized structure containing:

- Event ID
- Source log
- Raw timestamp
- Parsed UTC timestamp
- Extracted key-value details

3.5. Global Timeline Construction

All normalized events are merged into a single global timeline, sorted by parsed UTC timestamps. This timeline serves as the sole temporal reference for all analysis. Events without valid timestamps are excluded to avoid ambiguity.

3.6. RDP Session Reconstruction Methodology

RDP sessions are reconstructed using Windows session lifecycle semantics, not authentication heuristics.

3.6.1 Session Start

- Event ID 21 (LocalSessionManager)
- Treated as the only authoritative session start

3.6.2 Session End

- Event ID 24 (disconnect)
- Event ID 4634 (logoff)

3.6.3 Fallback Termination

- Applied only when no explicit termination is observed
- Clearly labelled as inferred termination
- Not treated as equivalent to a clean logoff

This design avoids session inflation and reflects actual Windows behaviour.

3.7. Temporal Correlation Using Grace Windows

Windows subsystems log events asynchronously. To account for this, the framework applies configurable grace windows:

- **GRACE_BEFORE**: captures preparatory actions
- **GRACE_AFTER**: captures delayed system changes

Events occurring within these windows are temporally associated with the nearest session and explicitly labelled as correlated rather than causal.

Each event is attached to only one session to prevent duplication.

3.8. DFIR Rule Engine

A deterministic DFIR rule engine operates on reconstructed sessions. Rules are transparent and map directly to extracted Event IDs.

Rule Name	**Event IDs**	**Severity**	**Description**
Brute Force Indicator	4625	High	Multiple failed RDP logons
User Account Created	4720	High	New local account
Privilege Escalation	4732	Critical	User added to administrators
Scheduled Task Persistence	4698, 129	High	Task-based persistence

Service Installation	7045	Critical	Service-based persistence
Log Clearing	1102	Critical	Anti-forensic behaviour
Short Session	—	Medium	Suspiciously short session
Long Session	—	Medium	Prolonged interactive access

Rules generate structured findings that include timestamps, severity, and explanation.

3.9. Machine Learning Module (Auxiliary)

The framework includes an optional machine learning module using Local Outlier Factor (LOF).

- Operates at the session level
- Uses features:
    - duration
    - authentication behaviour
    - persistence artifacts
    - event volume
- Disabled by default due to limited RDP session counts

ML results are contextual indicators only and do not override DFIR rules.

3.10. AI-Assisted Reporting (Auxiliary)

An optional AI-assisted reporting module uses the ChatGPT API to generate natural language explanations of validated DFIR findings.

Key constraints:

- No detection or correlation
- Evidence only explanations
- Analyst controlled activation

AI output never supersedes forensic findings.

3.11. Software Stack and Dependencies

- **Programming Language**: Python
- **Log Parsing**: EVTX parsing libraries
- **Analysis**: datetime, XML parsing
- **ML (optional)**: scikit-learn (LOF)
- **AI (optional)**: OpenAI API

4. Experimental Setup

This section describes the experimental environment and procedures used to evaluate the proposed framework. The goal of the experiment was to validate whether the framework can accurately reconstruct RDP sessions and correlate system level actions to those sessions using Windows Event Logs.

4.1. Test Environment

The experimental setup consisted of the following components:

- **Target system**: Windows 11 virtual machine
- **Virtualization platform**: VMware
- **Client system**: macOS host
- **RDP client software**: Microsoft Remote Desktop for macOS
- **Analysis mode**: Offline forensic analysis of exported Windows EVTX logs
- **Implementation language**: Python

The Windows 11 virtual machine was configured with default Windows logging settings. No additional logging or auditing policies were enabled beyond the standard configuration to reflect a realistic forensic scenario.

4.2. RDP Connection Procedure

The experiment followed a controlled sequence of actions to simulate realistic attacker behaviour:

1. An RDP connection attempt was initiated from the macOS system to the Windows 11 virtual machine using an incorrect password.
2. A subsequent RDP connection was established using valid credentials.
3. After successful authentication, interactive actions were performed during the RDP session, including:
    - Creation of a new local user account
    - Assignment of the new account to the local Administrators group
    - Installation of a new Windows service
    - Creation of a scheduled task

All actions were performed manually through the RDP session to simulate real attacker behaviour rather than automated malware execution.

4.3. Log Collection

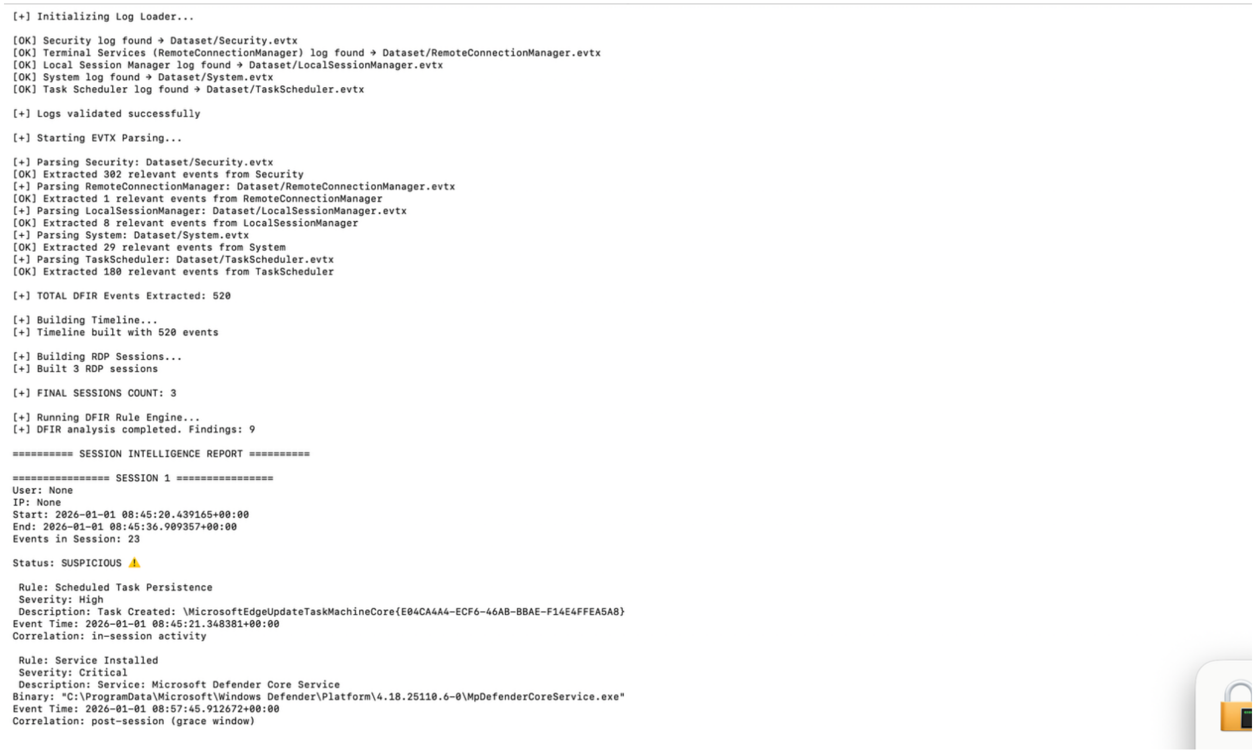
After completing the RDP activity, the following Windows Event Logs were exported from the target system for offline analysis:

- Security.evtx
- LocalSessionManager.evtx
- TerminalServices-RemoteConnectionManager.evtx
- System.evtx
- TaskScheduler.evtx

These logs were then supplied as input to the proposed framework.

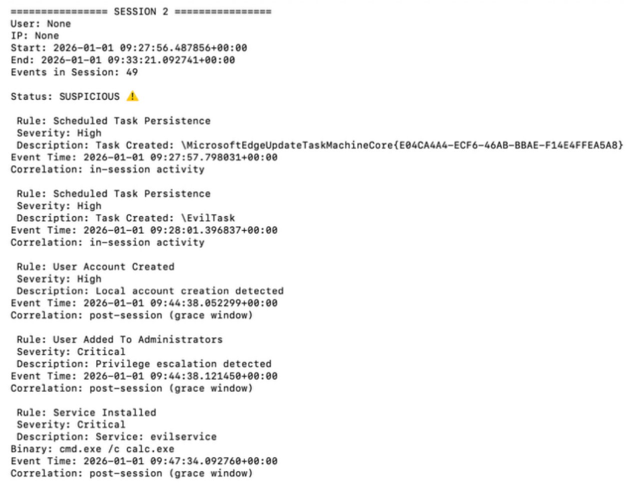
5. Evaluation and Results

This section presents the results of applying the framework to the collected logs. Screenshots of tool output, reconstructed timelines, correlated events, forensic analysis are included in this section.



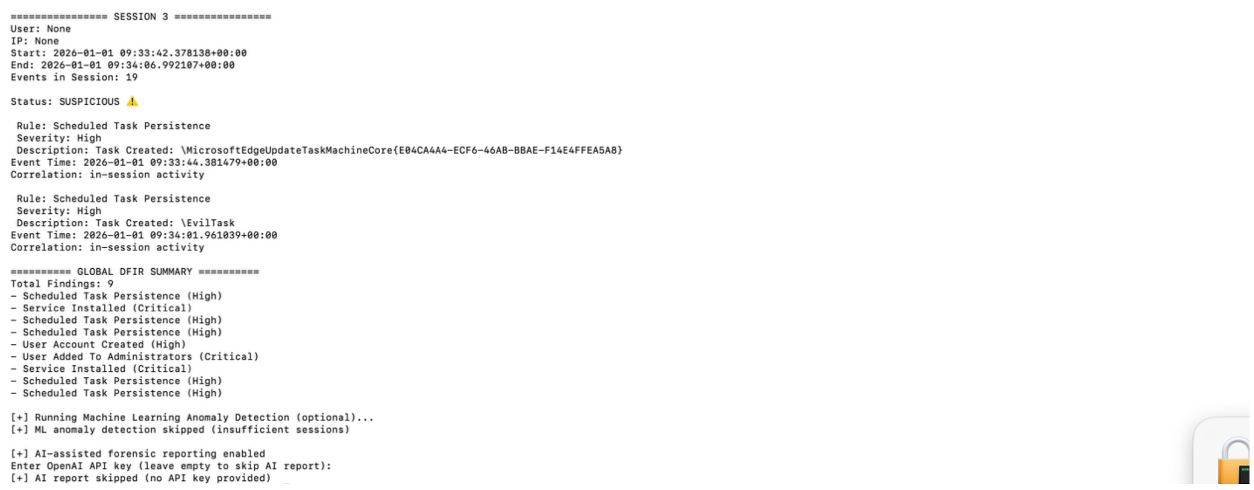
```
[+] Initializing Log Loader...

[OK] Security log found → Dataset/Security.evtx
[OK] Terminal Services (RemoteConnectionManager) log found → Dataset/RemoteConnectionManager.evtx
[OK] Local Session Manager log found → Dataset/LocalSessionManager.evtx
[OK] System log found → Dataset/System.evtx
[OK] Task Scheduler log found → Dataset/TaskScheduler.evtx

[+] Logs validated successfully

[+] Starting EVTX Parsing...

[+] Parsing Security: Dataset/Security.evtx
[OK] Extracted 302 relevant events from Security
[+] Parsing RemoteConnectionManager: Dataset/RemoteConnectionManager.evtx
[OK] Extracted 1 relevant events from RemoteConnectionManager
[+] Parsing LocalSessionManager: Dataset/LocalSessionManager.evtx
[OK] Extracted 8 relevant events from LocalSessionManager
[+] Parsing System: Dataset/System.evtx
[OK] Extracted 29 relevant events from System
[+] Parsing TaskScheduler: Dataset/TaskScheduler.evtx
[OK] Extracted 180 relevant events from TaskScheduler

[+] TOTAL DFIR Events Extracted: 520

[+] Building Timeline...
[+] Timeline built with 520 events

[+] Building RDP Sessions...
[+] Built 3 RDP sessions

[+] FINAL SESSIONS COUNT: 3

[+] Running DFIR Rule Engine...
[+] DFIR analysis completed. Findings: 9

========== SESSION INTELLIGENCE REPORT ==========

================ SESSION 1 ================
User: None
IP: None
Start: 2026-01-01 08:45:20.439165+00:00
End: 2026-01-01 08:45:36.909357+00:00
Events in Session: 23

Status: SUSPICIOUS ⚠️

 Rule: Scheduled Task Persistence
 Severity: High
 Description: Task Created: \MicrosoftEdgeUpdateTaskMachineCore{E04CA4A4-ECF6-46AB-BBAE-F14E4FFEA5A8}
Event Time: 2026-01-01 08:45:21.348381+00:00
Correlation: in-session activity

 Rule: Service Installed
 Severity: Critical
 Description: Service: Microsoft Defender Core Service
Binary: "C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.25110.6-0\MpDefenderCoreService.exe"
Event Time: 2026-01-01 08:57:45.912672+00:00
Correlation: post-session (grace window)
```

Figure 1



```
================ SESSION 2 ================
User: None
IP: None
Start: 2026-01-01 09:27:56.487856+00:00
End: 2026-01-01 09:33:21.092741+00:00
Events in Session: 49

Status: SUSPICIOUS ⚠️

 Rule: Scheduled Task Persistence
 Severity: High
 Description: Task Created: \MicrosoftEdgeUpdateTaskMachineCore{E04CA4A4-ECF6-46AB-BBAE-F14E4FFEA5A8}
Event Time: 2026-01-01 09:27:57.798031+00:00
Correlation: in-session activity

 Rule: Scheduled Task Persistence
 Severity: High
 Description: Task Created: \EvilTask
Event Time: 2026-01-01 09:28:01.396837+00:00
Correlation: in-session activity

 Rule: User Account Created
 Severity: High
 Description: Local account creation detected
Event Time: 2026-01-01 09:44:38.052299+00:00
Correlation: post-session (grace window)

 Rule: User Added To Administrators
 Severity: Critical
 Description: Privilege escalation detected
Event Time: 2026-01-01 09:44:38.121450+00:00
Correlation: post-session (grace window)

 Rule: Service Installed
 Severity: Critical
 Description: Service: evilservice
Binary: cmd.exe /c calc.exe
Event Time: 2026-01-01 09:47:34.092760+00:00
Correlation: post-session (grace window)
```

Figure 2



```
================ SESSION 3 ================
User: None
IP: None
Start: 2026-01-01 09:33:42.378138+00:00
End: 2026-01-01 09:34:06.992107+00:00
Events in Session: 19

Status: SUSPICIOUS ⚠️

 Rule: Scheduled Task Persistence
 Severity: High
 Description: Task Created: \MicrosoftEdgeUpdateTaskMachineCore{E04CA4A4-ECF6-46AB-BBAE-F14E4FFEA5A8}
Event Time: 2026-01-01 09:33:44.381479+00:00
Correlation: in-session activity

 Rule: Scheduled Task Persistence
 Severity: High
 Description: Task Created: \EvilTask
Event Time: 2026-01-01 09:34:01.961039+00:00
Correlation: in-session activity

========== GLOBAL DFIR SUMMARY ==========
Total Findings: 9
- Scheduled Task Persistence (High)
- Service Installed (Critical)
- Scheduled Task Persistence (High)
- Scheduled Task Persistence (High)
- User Account Created (High)
- User Added To Administrators (Critical)
- Service Installed (Critical)
- Scheduled Task Persistence (High)
- Scheduled Task Persistence (High)

[+] Running Machine Learning Anomaly Detection (optional)...
[+] ML anomaly detection skipped (insufficient sessions)

[+] AI-assisted forensic reporting enabled
Enter OpenAI API key (leave empty to skip AI report):
[+] AI report skipped (no API key provided)
```

Figure 3

5.1. RDP Session Reconstruction Results

The framework successfully reconstructed RDP sessions using LocalSessionManager Event ID 21 as the authoritative session start signal and Event IDs 24 and 4634 as termination signals.

The reconstructed sessions accurately reflected the timing and duration of the interactive RDP activity observed during the experiment. Session boundaries aligned with the actual connection and disconnection times observed during testing.

Fig. 1, 2, and 3 can illustrate the reconstructed session timelines.

5.2. Failed Authentication and Brute-Force Detection Behaviour

The initial incorrect password RDP attempt generated a failed authentication event in the Security log. However, this activity did not trigger a brute-force alert within the framework.

This behaviour is expected and intentional. The framework's brute-force rule requires multiple failed authentication attempts within a short time window to avoid false positives from single mistyped passwords or benign user errors. The absence of a brute-force finding in this case demonstrates a forensic aware rule design.

5.3. Correlation of Interactive Actions to RDP Sessions

All system level actions performed during the RDP session were successfully extracted and correlated to the reconstructed session timeline:

- **User account creation** events were detected in the Security log
- **Privilege escalation** via administrator group membership was identified
- **Service installation** events were extracted from the System log
- **Scheduled task creation** events were extracted from the TaskScheduler log

Some events occurred within the reconstructed session window, while others occurred shortly after session termination due to asynchronous Windows logging behaviour. These events were correctly associated with the session using the framework’s post-session grace window mechanism.

This demonstrates the importance of grace windows in accurately correlating delayed system events to active access.

5.4. Timeline Accuracy and Event Placement

The global timeline constructed by the framework preserved correct temporal ordering across all log sources. Events were not duplicated across sessions, and each DFIR relevant event was attached to a single session or explicitly labelled as correlated via grace windows.

This behaviour confirms that the framework avoids common pitfalls such as event duplication or ambiguous attribution.

5.5. Optional ML Module Observation

Machine learning based anomaly detection was automatically skipped due to an insufficient number of sessions, demonstrating correct safeguard behaviour.

AI-assisted forensic reporting was enabled but not executed due to the absence of an API key, confirming that the system remains fully functional in offline environments and that analytical augmentation is strictly optional.

6. Discussion

The experimental results demonstrate that the proposed framework can accurately reconstruct RDP sessions and correlate system level actions using Windows Event Logs alone. The experiment highlights several important observations:

- Session lifecycle events provide reliable session boundaries.
- Grace windows are necessary to account for asynchronous logging.
- Suspicious activity is interpreted relative to session context instead of being reported as standalone events.
- Conservative DFIR rules reduce false positives and produce accurate findings.

These observations reinforce the framework’s focus on forensic correctness.

7. Conclusion

This work presents a session centric DFIR framework for reconstructing and analysing RDP activity using Windows Event Logs. By combining timeline focused correlation with session inference, the tool presents forensic analysis that goes beyond isolated event analysis. The results demonstrate that meaningful RDP session reconstruction and explainable DFIR findings can be achieved using logs alone, supporting practical forensic investigation.

References

[1] J. Poling, "Windows RDP-Related Event Logs: Identification, Tracking, and Investigation," 2018. [Online]. Available: https://ponderthebits.com/2018/02/windows-rdp-related-event-logs-identification-tracking-and-investigation/.

[2] C. Marsden, "RDP Digital Forensics," May 2024. [Online]. Available: https://dfirinsights.com/2024/05/07/windows-remote-desktop-forensics/.

[3] F. Breitinger, H. Studiawan and C. Hargreaves, "SoK: Timeline based event reconstruction for digital forensics: Terminology, methodology, and current challenges," *Forensic Science International: Digital Investigation,* vol. 53, 2025.