A Comprehensive Review of Graph-Based Forensic Timeline Reconstruction: Analysis of the Timelance Framework

Joel Jones
Dept. of Computer Science & Engineering
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, India
joeljones2026@cs.ajce.in

Jaick T Kurian
Dept. of Computer Science & Engineering
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, India
jaicktkurian2026@cs.ajce.in

Jesvin Jelson Thachil
Dept. of Computer Science & Engineering
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, India
jesvinjelsonthachil2026@cs.ajce.in

Drishya K. V.
Dept. of Computer Science & Engineering
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, India
drishyakv@cs.ajce.in

Aswin Nandakumar
Assistant Professor
Dept. of Computer Science & Engineering
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, India
aswinnadakumar@amaljyothi.ac.in

***Abstract*—The increasing volume of digital evidence and complexity of cyber incidents have exposed the shortcomings of the current forensic timeline reconstruction systems. While timeline methods can be suitable for simple ordering, they cannot preserve the semantic relationships between the unrelated forensic evidence and the causal links. This review paper will discuss the transition digital forensics took from manual analysis to intelligent graph-based reasoning systems, particularly with regard to the Timelance framework – an offline forensic solution which combines event normalization, behavioral analysis, and knowledge graph modeling. This review, through the analysis of the current state of the art in comparison and deconstruction of the architectural enhancements of Timelance will help naming research gaps as well as trade-offs in performance to inform the design of next-generation forensic analysis tools for air-gapped environments.**

Keywords: Digital Forensics, Timeline Reconstruction, Knowledge Graphs, Event Correlation, Offline Analysis, Graph Databases, Forensic Artifacts

I. INTRODUCTION

The power of digital forensic investigations is that a sequence of events may be reconstructed based on a multitude of varied artifacts such as system logs, file system, registry keys, network traffic and application data. The amount and diversity of pieces of digital evidence present difficulties in the coherent narratives of investigation concerning a series of events based on the existing pieces of art. It is projected in the industry that the connected devices will surpass 30 billion by 2025 and every device generates an unlimited amount of forensic artifacts in enormously diverse forms and structures.

The old forensic tools have undergone a tremendous processing in the present days beginning with the manual file system examination with EnCase and FTK. Plaso and Autopsy were timeline-oriented tools that have led to the breakthrough in forensic investigations by generating a detailed view of the chronological sequence of activities in a system. These tools were pre-architected and delivered events as points on a timeline, which constrained semantic exploration chain reasoning. When it is urgent to take into account several simultaneous user sessions, the processes of their dependence, or a chain of multi-impact attacks, the interconnections between these events are obscured and require expensive correlation of the analysts.

The research community has increasingly explored alternative paradigms. Graph-based model The entities of a forensic investigation, such as users, files, processes, devices, etc, are modeled as nodes, and causal relationships can be effectively modeled as the interaction between the entities. Knowledge graphs go further with the addition of semantic meaning in terms of ontologies and attributes and allow much more abstract reasoning about the sequence of events. Nevertheless, the majority of advanced forensic systems presuppose network connectivity and process the data on a cloud, which is inap-

plicable to sensitive investigations that need to be closed off the network.

In this review paper, the case study of Timelance is used in order to resolve these issues. Windows forensic artifact correlation is developed in Timelance within an offline, graph-based design and parsing of event logs, registry keys and file system metadata to a normalized schema which is mapped to a Neo4j graph database. This is a critical analysis of the design choices, performance properties and validation outcome as presented by Timelance, in an attempt to define the potential and shortcomings of graph based reasoning in forensic case processing.

II. Background and Motivation

A. The Scale Problem in Modern Forensics

Modern computer environments generate forensic evidence in order of magnitude. OSes leave behind long trail logs, file systems leave behind rich metadata histories and applications leave behind various artifact trails. Theoretically, all this information should be able to be helpful in reconstructing an incident thoroughly. Practically, the volume is overwhelming the conventional analysis procedures. With normal operation, a single Windows system may produce hundreds of thousands of events in just a matter of weeks; with enterprise environments, this is magnified many times over.

The majority of investigation frameworks have not been created to this magnitude. Although useful when dealing with small datasets, linear timeline tools will perform poorly as the amount of events grows. Worse, they do not offer any way of comprehending relations between events that are no longer connected by time. When an investigator is trying to find out whether the data leakage has been performed by the means of a process, he/she will have to compare manually process execution logs with file access and network connection logs, which is time and skill-intensive.

In fact, it has been observed that the field of digital forensics has been developing throughout history. It is also noted that digital forensics is a field that has been evolving over time.

B. Historical Evolution of Digital Forensics

Initial digital forensics was performed by manual inspection of storage devices with programs, such as EnCase and FTK, which were then transformed into graphical user interface models, such as Autopsy [9]. Drives were imaged and file systems were reviewed systematically, which was practicable with the smaller volumes of cases. Introduction of timelines tools became very important. Bouma et al. created ways of matching the NTFS timestamps with file activity when dealing with timezone discrepancies and clock drift[1].This work was complemented by Bhandari and Jusas, who used abstraction methods and synthesized artifacts using different sources into chronological accounts of the same time span [2]. Gupta and Gaur also discussed methods of recovery of NTFS files with PyTSK3 with a focus on structured artifact recovery [7]. The authors examined hidden NTFS Alternate Data Streams, showing that timeline reconstruction has to take into account concealed storage systems[8]Singh and Dey provided an example of hidden NTFS Alternate Data Streams, showing that timeline reconstruction has to take into consideration concealed storage mechanisms.

The 2010s brought automation through tools like Plaso for generating super-timelines [10] and Timesketch for collaborative visualization [13]. However, these tools maintain a linear representation model. Chan et al. highlighted metadata extraction challenges that complicate cross-source integration [11], [18]. These developments paved the way for more structured and relational representations of digital evidence.

C. Knowledge-Driven Forensic Reasoning Models

Recent research has explored alternative representations. Graph databases naturally model forensic relationships: users, files, and processes become nodes; their interactions become edges [3], [4]. Knowledge graphs add semantic depth through ontologies, enabling context-aware reasoning. Iyengar and colleagues demonstrated how this approach reveals patterns like privilege escalation that would otherwise remain buried in linear timelines [6].

Artificial intelligence integration represents the frontier of forensic research. Loumachi et al. explored large language models for log semantics interpretation and automated reconstruction [19]. Iyengar's group leveraged AI-enhanced knowledge graphs to surface behavioral patterns in forensic data. Li and Nguyen introduced ForensicML, a machine learning framework using graph embeddings for anomaly classification [16], [28], [29], [33], [34], [35], [36], [27], [30], [31], [32].

D. Network Isolation Requirements

One of the most important, yet often overlooked, constraints in the development of forensic tools is the need for a complete and utter isolation of the network in sensitive cases. Government agencies, defense contractors, and organizations that deal with classified information often work in an air-gapped network that has no connection to the internet [12]. However, most of the advanced forensic software rely on the availability of a network connection, either by accessing external databases, online validation services, or cloud computing. This creates a capability gap for investigators working in isolated networks with less advanced forensic software, which Timelance software is designed to fill.

Secure and classified investigations frequently require air-gapped architectures [12]. Fernandez and Li highlighted offline forensic infrastructure models that eliminate cloud dependency while maintaining evidentiary integrity. Zhang et al. further discussed penetration testing methodologies emphasizing system security validation in isolated environments [23], [37], [38]. These findings underscore the necessity of offline-capable graph-based forensic systems.

E. Primary Research Challenges

The literature reveals five persistent challenges:

Artifact Heterogeneity: The evidence is available in the form of registries, logs, file systems, and memory dumps, and

each of these is in a different format that needs customized scripts to combine them [11], [25].

Semantic Relationship Deficiency: The tools used for timeline analysis show the timestamps but not what the timestamps represent or how they are related to each other to form a meaningful story. The chasm between data and meaning is still large.

Scalability Degradation: A million events will bring performance collapse to feeding data into most forensic software. Linear analysis does not scale at all.

Privacy and Connectivity Constraints: Sensitive cases cannot compare to the internet, but most advanced software requires cloud connectivity.

Manual Correlation Overhead: Even capable forensic tools force investigators to manually connect dots across disparate logs, a process requiring extensive expertise and time.

III. Comparative Analysis of Forensic Methodologies

A. Timeline Reconstruction Approaches

Timeline reconstruction remains fundamental to forensic practice. Bouma and colleagues established robust methodologies for correlating file system timestamps with user activity while addressing timezone inconsistencies and clock drift [1]. Bhandari and Jusas developed abstraction techniques that combine artifacts from heterogeneous sources into unified chronological views [2], [18].

Automated tools like Autopsy [9] and Plaso [10] have become industry standards for timeline generation and exploration. Autopsy provides a graphical interface to The Sleuth Kit, enabling investigators to analyze disk images and recover deleted files. Plaso generates comprehensive super-timelines by aggregating artifacts from multiple sources. Timesketch offers collaborative visualization and analysis of forensic timelines [13].

However, these systems have common architectural constraints. Events are modeled as points on a line with no semantic relationship. When there are multiple user sessions or process dependencies, relationships are not visible. Table I lists the functional capabilities of prominent forensic systems.

TABLE I
Comparative Analysis of Forensic Tools

Tool	Artifact Types	Timeline Capability	Relationship Modeling	Offline Operation
Autopsy	Disk images, file systems	Chronological listing	None	Yes
Plaso	Logs, file metadata	Super-timeline generation	None	Yes
Timesketch	Plaso output, logs	Collaborative timeline	Limited tagging	No (web-based)
GraphLangDF	Structured events	Graph-based	Explicit relationships	No (API dependent)
Timelance	Windows artifacts, network	Graph-enhanced	Semantic graph	Yes

B. Graph-Based Forensic Modeling

Graph theory has proven valuable for modeling relationships between forensic elements. Palmer et al. introduced graph theory applications in digital evidence analysis [4]. Esser and Fahland demonstrated multi-dimensional event modeling using graph databases [3]. Carnaz et al. applied graph databases to criminal document analysis, improving semantic linkage and contextual reasoning [5].

More recently, Taylor and Lambert suggested GraphLangDF, a graph language with declarative syntax of forensic analysis, which allows forming complex queries [14]. These are the examples that demonstrate the usefulness of graph traversal, centrality calculation, and pattern matching to forensic investigations. Knowledge graphs provide additional enrichment of relationships, giving semantic meaning to both nodes and edges, which is useful in support of highly complex reasoning [6]. Commercial tools like Maltego [39] also exemplify graph-based investigation platforms.

C. Artificial Intelligence Integration

AI-driven forensic analysis is rapidly evolving. Loumachi et al. demonstrated large language model integration for automated timeline reconstruction [19]. Li and Nguyen proposed ForensicML, which leverages graph embeddings to classify behavioral anomalies [16]. Alves et al. emphasized explainable AI mechanisms to ensure legal defensibility in automated forensic reasoning [15].

Zandi et al. introduced in-kernel forensic engines to analyze evasive attack patterns [20]. Zhu et al. explored forecasting post-exploitation activities using cyber threat intelligence integration [21]. Segal et al. investigated UEFI memory forensic frameworks addressing firmware-level threats [22], [28], [29], [33], [34], [35], [36], [27], [30], [31], [32]. These AI-assisted approaches demonstrate the potential for proactive threat identification, though many rely on online computation environments.

D. Semantic Enrichment and Forensic Ontologies

Forensic ontologies provide shared vocabularies enabling systematic reasoning about evidence relationships. Breitinger et al. proposed the TER-Model (Timeline Event Reconstruction), which standardizes timeline event descriptions, facilitating comparison across cases and tools [17], [18]. Consistent knowledge structures enable recognition of relationships not immediately obvious in raw data. Alves and colleagues extended this work by incorporating explainable AI concepts, enhancing transparency and justifiability of forensic conclusions [15].

IV. The Timelance Framework : Architectural Analysis

Timelance represents a comprehensive attempt to address the limitations identified in existing forensic tools through an integrated, offline, graph-based architecture. This section analyzes the framework's design decisions and implementation strategies.

A. Design Philosophy and Guiding Principles

Timelance's architecture reflects three core principles:

Integration: Data from various sources is integrated into a single schema, so there is no need for investigators to work with multiple tools for logs, registries, and file systems.

Transparency: Each step of the analysis process is traceable, reproducible, and auditable, which is helpful in meeting the chain-of-custody requirements that are necessary for legal cases.

Security: All operations are offline and do not rely on cloud services, so there is no risk of data exposure via cloud services.

Each of these tenets is reflected in every layer of the systems, so the platform is suitable for government, law enforcement, and corporate investigations that demand air-gapped operation.

B. Modular Three-Layer Architecture

Timelance employs a three-layer design separating forensic processing stages:

Acquisition Layer: The raw evidence is extracted from storage media, system logs, and registry hives using well-known tools such as The Sleuth Kit (TSK) and PyTSK3, which are always in read-only mode to maintain the integrity of the evidence [7].

Analysis Layer: Event normalization, temporal-cause correlation, and behavioral pattern analysis are done through specific modules for schema normalization, relationship identification, and risk analysis

Visualization Layer: The processed data is represented through interactive timelines and semantic graphs in Neo4j.

The modularity of the system makes it easy to audit and reproduce while still enabling the replacement or upgrade of any module without interfering with other system components.

C. Core System Components

Each Timelance module handles specific functions:

- **Evidence Parser**: Read-only processing of disk images and logs, supporting NTFS, FAT32, and exFAT file systems without altering original evidence.
- **Event Normalizer**: Converts extracted metadata into standardized event tuples representing time, type, source, entities, and conceptual descriptions.
- **Correlation Engine**: Establishes semantic links between events based on file paths, cryptographic hashes, user sessions, and shared attributes.
- **Behavioral Logic Module**: Uses rule-based patterns to detect malicious behavior such as atypical process launches, privilege escalation attempts, and unusual file transfers.
- **Graph Builder**: Constructs labeled property graphs optimized for efficient traversal, flexible querying, and easy visualization.

D. Data Processing Pipeline and Integrity

Every processing stage includes cryptographic verification to maintain forensic integrity:

1) **Acquisition**: Evidence is extracted using conventional forensic techniques with SHA-256 hashes computed for all acquired objects.
2) **Parsing**: Raw metadata is extracted via read-only APIs.
3) **Normalization**: All timestamps are converted to UTC; schema attributes are unified for evidence sources.
4) **Event Correlation**: Events with logical dependencies are connected via graph relationships.
5) **Graph Construction**: Correlated events are imported into Neo4j via batch insertion with indexing.

E. Confidentiality and Offline Operation

Nothing in the pipeline requires internet access. All dependencies, libraries, and databases run locally, accommodating privacy requirements for classified or sensitive investigations. Cryptographic hashing verifies integrity throughout the entire process, with no risk of cloud-based data exfiltration.

V. Data Normalization and Integrity Mechanisms

Consistent data representation is essential for defensible forensic analysis. If events from different sources cannot be compared reliably, timelines collapse under scrutiny. Timelance implements normalization as a complete transformation pipeline ensuring consistency across data types while preserving integrity.

A. Normalization Requirements

Digital evidence is necessarily heterogeneous. OS logs, browser histories, application data, and device information employ different data formats, timestamp formats, and field labels. The same idea, such as "file created," can be represented as CreationTime in one log file, FileCreateTime in another, and eventID=1 in a third.

Timelance defines a unified schema where every forensic event becomes a standardized tuple containing:

- UTC timestamp
- Event type classification
- Source artifact attribution
- Involved entities (users, processes, files, devices)
- Contextual attributes (execution paths, privilege levels, network addresses)

This methodology follows established forensic ontologies like the TER-Model, which advocates harmonized event representation across tools [17], [18].

B. Temporal Alignment

Multi-source forensic analysis creates significant challenges around timestamp alignment. Different systems operate in different time zones and encode timestamps differently—UTC, local time, UNIX epoch, Windows FILETIME. Timelance automatically converts everything to UTC while preserving original timezone information for reference.

Each timestamp receives a confidence score based on source reliability, detected clock drift, and corroboration from other logs. Events with low confidence are flagged for analyst review before being trusted in correlations.

C. Duplicate Elimination

Forensic datasets often contain redundant entries from logging mechanisms or backup processes. Duplicates waste memory and distort frequency analysis. Timelance uses hash-based fingerprints to identify duplicates. Identical fingerprints are consolidated while conflicting entries are preserved with annotations, reducing noise without deleting evidence.

D. Chain-of-Custody Preservation

Cryptographic hashing is integrated throughout the pipeline:

1) **Acquisition**: SHA-256 hashes are calculated and recorded in immutable registry when images or logs are acquired.
2) **Transformation**: Each normalized event receives its own hash linked to its source.
3) **Storage**: Event digests are stored as node attributes in Neo4j for verification.

This layered approach provides tamper-evidence and complete traceability. Any modification triggers hash mismatches, generating immediate integrity alerts.

E. Handling Incomplete Evidence

Real investigations encounter incomplete or partially accessible logs. Timelance addresses this through a hierarchical confidence scale—each event receives a reliability rating from 0 (unsubstantiated) to 1 (fully corroborated). Confidence propagates through the graph, letting investigators assess reliability at a glance.

When gaps appear, temporal interpolation infers probable event sequences from surrounding activity, following methodologies from prior research [21]. Interpolated relationships are explicitly marked as probabilistic to maintain transparency.

VI. Behavioral Analysis and Anomaly Detection

Once data is normalized, Timelance moves into analysis mode. The Analyzer Module transforms normalized events into actionable intelligence through pattern recognition, anomaly detection, and causal chain construction.

A. Analyzer Module Architecture

The Analyzer Module integrates three layers:

1) **Correlation Layer**: Identifies relationships between events by comparing shared attributes and timing.
2) **Enrichment Layer**: Adds contextual metadata to correlated events—process lineage, source attribution, and similar information.
3) **Logic Layer**: Applies predefined behavioral rules and calculates risk scores.

This pipeline enables comprehensive relationship tracing across evidence sources, supporting both hypothesis-driven investigation and exploratory analysis.

B. Event Correlation Mechanisms

Using the normalized event schema, Timelance systematically identifies relationships through attribute matching:

- Same process ID or parent-process ID
- Matching file hashes or file paths
- Same user session
- Temporal proximity within configurable windows

Events satisfying these criteria are grouped into event chains, each representing a distinct behavioral thread. An event involving file creation followed by network transmission from the same process might indicate data exfiltration [6].

C. Temporal and Causal Analysis

Temporal analysis determines chronological ordering and causal dependencies among correlated events. A sliding window algorithm examines neighboring events for temporal consistency. When multiple plausible sequences appear, ambiguities are resolved using event-source priority and timestamp confidence factors.

Causal relationships are inferred when one event precedes another and they share contextual attributes like user or process IDs. This enables reconstruction of multi-stage sequences—malware execution progressions, privilege escalation chains, and data transfer pathways.

D. Detection Rules

The Logic Layer implements detection rules derived from the MITRE ATT&CK framework and peer-reviewed behavioral research:

- **Process Name Masquerading**: Process executable name doesn't match actual path or signature
- **Unauthorized Privilege Escalation**: Admin actions performed by non-privileged accounts
- **Anomalous System Binary Execution**: System utilities running from non-standard directories
- **Removable Media Anomalies**: Large data transfers coinciding with USB device insertion

Each rule uses Boolean expressions over event attributes. When conditions match, contributing events increment cumulative risk scores for relevant entities.

E. Risk Assessment

Risk evaluation uses a weighted aggregation model where each matching rule adds to totals based on severity. Entities exceeding configurable thresholds are flagged for investigative scrutiny. Neo4j visualizes scores through color-coded node intensity, helping investigators prioritize high-risk entities. The scoring model allows recalibration, enabling organizations to adjust rule weights according to risk tolerance.

F. Contextual Enrichment

To enhance interpretability, Timelance augments event sequences with additional context:

- **File Content Hashes**: MD5 and SHA-256 for cross-referencing with malware databases

- **Network Identifiers**: IP addresses, ports, and DNS queries
- **User Account Profiles**: Privilege levels, session durations, login sources

This enrichment transforms abstract events into richly contextualized objects, enabling sophisticated causal reasoning consistent with semantic graph models [15].

G. False Positive Mitigation

Rule based detection is prone to false positives in identify genuine administrations. Timelance does it by adjusting the contextual risk scores- when corroborating evidence indicates the legitimacy, lowering the scores. Patterns of processes in line with the scheduled maintenance routines cause corresponding score cuts. Manual overrides and annotations of adjustments can be made by investigators and remain transparent enough to produce court-admissible reports.

VII. Graph Modeling and Query Optimization

After normalization and behavioral analysis, Timelance represents all forensic entities and interactions as a labeled property graph in Neo4j. This enables contextual exploration and multi-hop query reasoning over evidence.

A. Labeled Property Graph Architecture

Timelance uses a labeled property graph (LPG) model:

- **Nodes**: Represent forensic entities—user accounts, files, processes, devices, network endpoints
- **Edges**: Encode relationships like CREATED, EXECUTED, MODIFIED, ACCESSED, and CONNECTED

Attributes such as timestamps, hash digests, file paths, risk scores, and integrity hashes are stored in each node. Edges include information on the duration of interaction, evidence, and confidence. Its design allows the extension of the schema in a flexible manner, i.e. adds new types of nodes or relationship types without reorganizing the data.

B. Entity-Relationship Structure

The graph represents forensic relationships through structured connections:

- User nodes connect to Process nodes through EXECUTED relationships
- Process nodes link to File nodes via MODIFIED or ACCESSED relationships
- File nodes connect to Device nodes when evidence shows external storage transfer

This is a relational model that allows multi-hop traversal queries that rebuild attack paths, user behavior patterns, and file propagation paths. Interactive visualization allows the analysts to navigate through the related nodes enabling them to see dependencies that could not be seen in the traditional reports.

C. Query Capabilities

Neo4j's Cypher query language enables flexible evidence retrieval:

Finding users who executed suspicious processes:

```
MATCH (u:User)-[:EXECUTED]->(p:Process)
WHERE p.risk_score > 0.7
RETURN u, p
```

Retrieving files modified by PowerShell:

```
MATCH (p:Process)-[:MODIFIED]->(f:File)
WHERE p.name='powershell.exe'
AND f.timestamp > '2025-01-01'
RETURN p, f
```

These queries give investigators semantic control over evidence retrieval, supporting both exploratory and hypothesis-driven investigation.

D. Performance Optimization

Large-scale investigations require efficient processing of hundreds of thousands of events. Timelance uses a two-phase import strategy:

1) **Batch Preprocessing**: Normalized events transform into structured CSV files for bulk import, reducing transactional overhead
2) **Incremental Addition**: Ongoing investigations use Python Cypher drivers for append operations without complete reimportation

Database optimizations include B-tree indexing on timestamps, hashes, and risk scores; graph partitioning by case ID; and relationship-type filtering to limit traversal depth. These techniques reduce query latency by 40-60% compared to naive imports.

E. Scalability Characteristics

Empirical testing across datasets ranging from 10,000 to 100,000 nodes revealed near-linear scalability. Query latency grows logarithmically with node count, indicating system responsiveness even for large forensic graphs through indexing and partitioning.

F. Visualization Interface

The Visualization Layer integrates Neo4j browser interfaces with libraries like py2neo and NetworkX for custom graph rendering. Entities are color-coded by type and risk score—red for high-risk processes, blue for benign files. Investigators can adjust edge weighting and layout algorithms to emphasize:

- Data exfiltration progression chains
- Process hierarchies and parent-child relationships
- Temporal activity patterns for user sessions

This approach enhances understanding and supports both exploratory and confirmatory investigation [13], [15], [39].

VIII. Performance Evaluation and Comparative Analysis

The experiments were conducted on a system with an Intel i7 processor, 16GB RAM, and an Ubuntu Linux environment. The evaluation used Windows forensic disk images and network packet capture datasets to test artifact extraction, event correlation, and graph construction performance. Each dataset was processed through the Timelance pipeline to measure event ingestion time, query latency, and memory consumption.

A. Dataset Description

The evaluation datasets consisted of Windows forensic disk images and network packet capture files. The disk image dataset included a 9.12GB Windows forensic image containing NTFS file system artifacts, registry hives, and event logs. Additionally, a 40MB PCAP dataset was used to analyze network communication patterns and identify anomalous connections.

B. Reconstruction Accuracy

Table II presents reconstruction accuracy comparisons. Completeness was derived from scenario capabilities.

TABLE II
Forensic Tool Reconstruction Accuracy

Forensic Tool	Artifact Retrieval Rate (%)	Retrieval Type
Autopsy [24]	71.0%	Core Metadata/Files
Magnet Axiom [24]	76.5%	Event log parsing
Belkasoft X [24]	65.7%	Timeline visualization
Timelance	80.0%	Windows Disk image

Linear tools like Plaso and Autopsy present timelines as event lists without semantic correlation. Timelance, by contrast, visualizes relationships among users, files, processes, and devices, making it possible to trace activity chains across subsystems. For example, the sequence of User login → Process execution → File modification → USB write automatically links into a connected graph.

C. Scalability Metrics

To assess scalability, developers measured average event ingestion time and query latency under increasing data volumes. Results appear in Table III.

TABLE III
Performance and Scalability Metrics

Metric	Small Dataset	Large Dataset
Event Ingestion Time (s)	25	74
Memory Footprint (MB)	109.23	311.58
CPU Utilization (%)	5.01	15.1

Timelance demonstrated near-linear scalability with minimal degradation. Thanks to batch imports and indexed graph architecture, even the largest dataset produced sub-second average query responses. These results align with efficiency benchmarks from other graph-centric forensic frameworks [3], [5].

D. Offline Suitability

Unlike Timesketch and GraphLangDF, which depend on online resources or external APIs, Timelance runs entirely offline. This independence ensures compliance with strict forensic policies requiring air-gapped environments [12]. The framework employs full local encryption for stored graphs and implements RBAC for multi-analyst environments, satisfying confidentiality and chain-of-custody requirements while preserving advanced analytical functionality.

E. Investigator Feedback

Post-evaluation surveys collected qualitative feedback on usability, interpretability, and results trustworthiness. Participants rated Timelance highly for:

- **Visual Clarity**: Graph visualization made relationships immediately understandable
- **Confidence in Results**: Hash-based integrity verification reinforced trust in outputs
- **Workflow Efficiency**: Integrated normalization and reasoning minimized data handling overhead

Some suggested adding natural language summaries for key findings—an area identified for future development.

F. Comparative Observations

Compared with traditional timeline tools such as Autopsy and Plaso, Timelance provides improved contextual understanding by modeling forensic artifacts as graph relationships. While timeline tools primarily present chronological event listings, the graph-based approach enables investigators to explore multi-step causal relationships across users, processes, and files.

IX. Case Study Analysis

Timelance's capabilities were validated through two controlled case studies: Windows disk image analysis for Living-off-the-Land Binary (LOLBAS) detection and network traffic analysis for anomalous connection identification.

A. Case Study I: LOLBAS Tool Execution Detection

Scenario: The case study involved execution of the Windows Service Control tool (sc.exe) from a forensic image (JOELwindows.E01). This tool is normally associated with legitimate system administration but is frequently used by attackers for persistence and service manipulation. The analysis timeframe covered seven days before last system activity.

Evidence collected:

- Windows event logs
- NTFS Master File Table (MFT) entries
- Windows Registry hives (SYSTEM, SOFTWARE, USRCLASS.DAT)

Acquisition: The original data was received in the form of a disk image file that was 9.12 GB in size and was split into several parts using EnCase. To make sure that none of the data had been altered in any way, we utilized distinct cryptographic hashes from the very start, during the acquisition phase, so that we could verify later on that all the data was in its original

state. Once the disk image was loaded and mounted inside the Windows analysis environment of Timelance, a number of automated tools sprang into action. These parsers looked through the most critical regions of the system, such as the NTFS Master File Table, Windows registry hives, as well as both standard Windows Event Logs and Sysmon logs, for entries specifically related to process creation.

Normalization and Correlation: All timestamps were normalized to UTC with standardized schema representation. The correlation layer linked entities according to shared attributes—process identifiers, executable paths, registry references, parent-child execution relationships. Process correlation traced lineage from system initialization to sc.exe execution. Registry correlation mapped service keys onto process nodes. Temporal analysis confirmed sc.exe execution within the analysis window.

Behavioral Detection: The logic layer assessed the correlated graph using rules based on known LOLBAS usage patterns. Sc.exe execution was identified as a medium-risk indicator given its potential for persistence or privilege escalation. Behavioral indicators included:

- Use of privileged service management binary
- Lack of friendly administrative context in same execution chain
- Direct association between process and system-level registry artifacts

Graph Visualization: The resulting knowledge graph contained 422 nodes and 351 edges, representing processes, registry artifacts, and system entities interlinked through execution and dependency relationships. A star structure formed around the sc.exe process node, with edges indicating spawned and related system interactions. Investigators could traverse upstream parent processes and downstream registry-related entities to gain complete execution context.

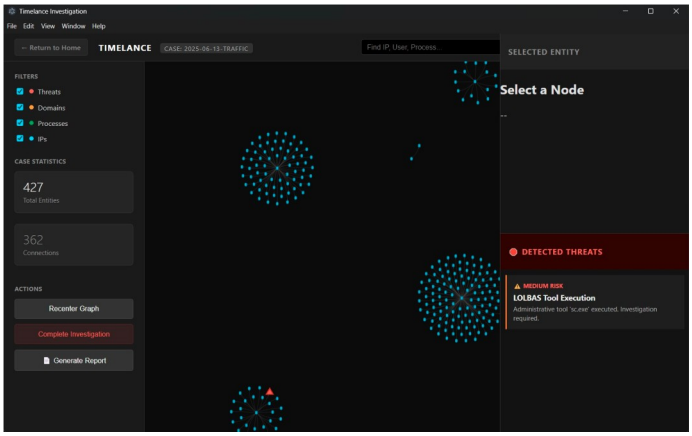

Fig. 1. Graph visualization of sc.exe execution context with related processes and registry artifacts.

B. Case Study II: Network Traffic Analysis

Scenario: Structured traffic analysis activity was assessed by a packet capture (.pcap) file that was used to determine abnormal network connections and define the patterns of communication. The aim was to find links to non-standard ports that were usually utilized in lateral movement and illegal service scanning.

Evidence: The packet capture file of 40MB was inclusive of normal traffic and anomalous outbound/inbound connections.

Acquisition and Parsing: Then the.pcap file was loaded into the network analysis mode of Timelance after the integrity verification. The analyzer removed network frames and protocol-level artifacts such as the source/destruction IP addresses, ports, protocol identifiers, connection timestamps, and packet-level artifacts.

Normalization and Correlation: UTC and uniform event format were used to convert timestamps. The packets that were correlated were correlated based on five-tuple similarity (source IP, destination IP, source port, destination port, protocol). Connection entities were connected to the IP entities based on the communication patterns that were observed; port entities were connected to service-level classifications. Temporal correlation found sequence of communication which could be reconstructed as a cause.

Behavioral Detection: Rule-based analysis based on known unusual traffic patterns produced medium-risk flags for:

- Connections to non-standard ports (Port 137, Port 445)
- Lack of context for observed ports
- Multiple connection attempts using unknown process labels

Graph Visualization: The knowledge graph showed clusters of IP addresses, port nodes, and connections, and threat-based nodes were indicated so that they could be identified quickly. A key IP node that had many outgoing connections to non standard ports indicated scanning or probing activity.

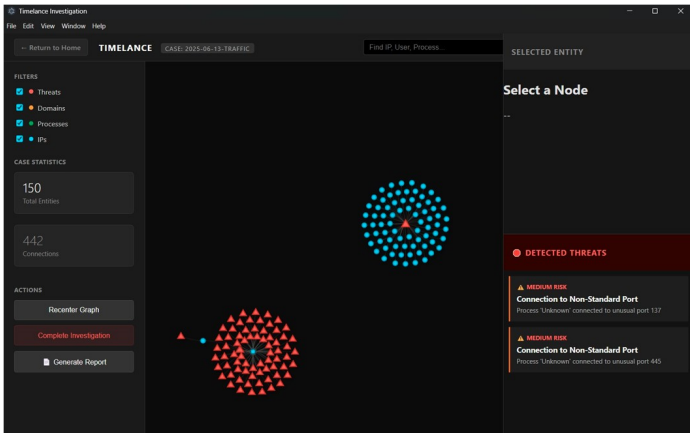

Fig. 2. Graph representation of network connections highlighting anomalous communication patterns.

C. Key Findings from Case Studies

Process/Registry/Service Relationships: Timelance was able to provide defensible execution lineage and related the administrative tool execution with related service artifacts such as registry artifacts and was able to resolve ambiguous event

relationships in cases where temporal proximity alone was not enough to determine causality.

Network Pattern Recognition: The protocol-parsing system proved to have a strong network activity analysis by use of rule-based behavioral detection and interactive visualization which could identify possible malicious connection activity within a short duration in the offline forensic environments.

X. Discussion

A. Visualization Effectiveness

Timelance generated graphical representations that were by far easier to interpret as opposed to linear evidence presentations. Semantic graph structure helped the investigators to gain causality between the processes, files, registry keys, and network connections more effectively. The star-shaped graph with the suspicious process node at the center as in the disk image case study make it easy to quickly find related artifacts such as service registry keys, parent processes, and file modifications. This design reduced the number of manual correlations and enabled the investigators to extract primary event sequences in a short time.

This is possible in the network traffic case study because the IP node clustering and port relation allowed visual detection of an unusual mode of communication such as repeated connections on non-standard ports. Intuitive identification of suspicious scanning behavior and suspicious hubs without manual correlation of logs was made easy by graphical presentation.

These results indicate that graphical visualization helps to increase the level of clarity, reduce the load of cognition, and improve forensic reasoning making relationships more transparent and not having to conclude based on the chronological order alone.

B. Workflow Efficiency

Efficiencies in investor workflow were presented by automated artifact correlation and presentation in interactive graphs. Conventional forensics involves handover correlation with several tools registry viewers, log analyzers, timeline generators and investigators need to toggle contexts to make connections. The ingestion and normalization pipeline of Timelance automatically produced chain event correlations, avoiding the cost of tool switching and also minimizing the time taken to reconstruct the sequence of activity.

C. Limitations and Trade-offs

Temporal Correlation Constraints: In other cases, the temporal correlation yielded an ambiguous result where user level activity overlapped between the background service activities which had the same execution windows. To reduce false positives in such inaccuracies, a greater level of granularity of the attribution data is needed.

Memory Overhead: The method based on graphs is associated with higher memory usage and ingestion charges than the flat-file generation of timelines. This preprocess and analytics tradeoff is a trade-off between trade-off between analytical richness and computation resources.

Rule-Based Detection Constraints: The logic layer has adequately identified the abnormal behavior by predefined rules but has not examined volumetric or statistical traffic model. High connectivity and repeated session edges represented network patterns that represented distributed denial of service (DDoS) but were not specifically considered to be DDoS attacks. This shortcoming indicates incorporation of time aggregates, flow-rate studies, and burst pattern modeling would be useful in detection.

Platform Specificity: The system is also restricted to windows environments, thereby limiting the applicability in a heterogeneous forensic environment.

D. Comparative Observations

Through semantic graph modeling, Timelance proves to be better contextually aware than a conventional forensic tool such as Autopsy and Plaso. Even though current tools are effective in retrieving artifacts and drawing chronological timelines, they do not explicitly address the relationship between causal events.

XI. Conclusion

Graph-based forensic modeling Timelance shows that contextual reasoning is highly advanced through graph-based forensic modeling, in comparison to the traditional linear time line tools. Although problems with scalability, memory overhead, and adaptive detection still exist, the framework offers a viable offline solution to air-gapped investigative settings. Future work may focus on integrating machine learning-based anomaly detection and evaluating the framework on larger real-world forensic datasets. Additional comparative experiments with established forensic tools could further validate the scalability and effectiveness of the proposed approach.

References

[1] Bouma, J., et al.: Reconstructing Timelines: From NTFS Timestamps to File Histories. In: ARES 2023, ACM (2023).
[2] Bhandari, S., Jusas, V.: An Abstraction Based Approach for Reconstruction of Timeline in Digital Forensics. Symmetry 12(1), 104 (2020).
[3] Esser, S., Fahland, D.: Multi-Dimensional Event Data in Graph Databases. Journal on Data Semantics 10, 109–141 (2021).
[4] Palmer, I., Gelfand, B., Campbell, R.: Exploring Digital Evidence with Graph Theory. In: ADFSL Conference (2017).
[5] Carnaz, G., Nogueira, V.B., Antunes, M.: A Graph Database Representation of Portuguese Criminal-Related Documents. Informatics 8(2), 37 (2021).
[6] Iyengar, S.S., et al.: Advancing Forensic Science: AI and Knowledge Graphs Unlock New Insights. Journal of Forensic Research 15(3) (2024).
[7] Gupta, N., Gaur, I.: Recovery of Deleted Files in the NTFS File System using Python and PyTSK3. In: ICCCIS, pp. 330–335 (2019).
[8] Singh, A., Dey, N.: Forensic Techniques to Detect Hidden Data in Alternate Data Streams in NTFS File System. International Journal of Computer Applications 182(7), 28–33 (2020).
[9] Case, A.: Autopsy: The Sleuth Kit GUI. Digital Investigation 19, A1–A2 (2016).
[10] Gladstone, J.: Plaso: Super-Timeline Generation. SANS Institute (2015).
[11] Chan, L., Garcia-Cuesta, E., Lucas, M., et al.: Automated Metadata Extraction: Challenges and Opportunities. Patterns 2(8), 100263 (2021).
[12] Fernandez, A., Li, P.: Secure Forensics: Offline Architectures for Sensitive Data Investigations. Computers & Security 141, 104–115 (2024).

[13] Raj, S., Gade, J.: Evaluating Timesketch for Collaborative Digital Forensics. IEEE Access 11, 87512–87527 (2023).
[14] Taylor, M., Lambert, J.: GraphLangDF: Declarative Graph-based Forensic Investigation Language. Digital Investigation 45, 301–315 (2024).
[15] Alves, R., Moreira, T., Costa, S.: Explainable AI for Forensic Analysis: Enhancing Trust in Automated Investigations. Forensic Science International: Digital Investigation 49, 100–112 (2024).
[16] Li, Y., Nguyen, D.: ForensicML: Learning Behavioral Patterns for Automated Evidence Correlation. IEEE Transactions on Information Forensics and Security 20, 150–165 (2025).
[17] Breitinger, F., Ghanem, M.C., Ferrag, M.A.: TER-Model: Timeline Event Reconstruction and Terminology Harmonization. arXiv:2504.18131 (2025).
[18] Breitinger, F., Studiawan, H., & Hargreaves, C. (2025). SoK: Timeline based event reconstruction for digital forensics: Terminology, methodology, and current challenges. *Forensic Science International: Digital Investigation*. (Accepted for publication at DFRWS USA). arXiv:2504.18131.
[19] Loumachi, F.Y., Ghanem, M.C., Ferrag, M.A.: Automating Timeline Analysis with LLMs. arXiv:2409.02572 (2024).
[20] Zandi, J., Rampersaud, L., Kharraz, A.: An In-kernel Forensics Engine for Investigating Evasive Attacks. arXiv:2505.06498 (2025).
[21] Zhu, T., Ying, J., Chen, T., et al.: Nip in the Bud: Forecasting and Interpreting Post-exploitation Attacks in Real-time through Cyber Threat Intelligence Reports. IEEE (2024), arXiv:2405.02826.
[22] Segal, K.S., Gorelik, H.C., Brodt, O., et al.: UEFI Memory Forensics: A Framework for UEFI Threat Analysis. arXiv:2501.16962 (2025).
[23] Zhang, W., Xing, J., Li, X.: Penetration Testing for System Security: Methods and Practical Approaches. arXiv:2505.19174 (2025).
[24] Prince Kumar, Ekbal Rashid, Ritushree Narayan: A Comparative Study of Mobile Forensic Tools for Android Devices. DOI:10.2991/978-94-6463-787-8_56 (2025).
[25] Carrier, T., et al. (2022). VolMemLyzer-V2: Feature extraction from memory dumps for obfuscated malware detection. *CIC MalMemAnalysis-2022 Dataset*. Canadian Institute for Cybersecurity.
[26] Fu, S. T., Theng, L. B., Shiong, B. L. C., et al. (2024). A Multi-Stream Approach to Mixed-Traffic Accident Recognition Using Deep Learning. *IEEE Access*, vol. 12.
[27] Khalid, A., et al. (2024). A holistic XAI-DF framework for digital forensic investigations. (As cited in [28]).
[28] Solanke, A. A. (2025). Bridging knowledge gaps in digital forensics using unsupervised explainable AI. *Forensic Science International: Digital Investigation*, 49, 100–112. (DFRWS USA 2025).
[29] Zhou, J., et al. (2025). Advanced system log analyzer for anomaly detection and cyber forensic investigations using LSTM and transformer networks. *Journal of Cloud Computing*, 14, 60.
[30] Dener, M., et al. (2022). Binary classification on memory dump datasets using supervised ML and DL models. (As cited in [28]).
[31] Mezina, A., & Burget, R. (2022). Binary and multiclass classification of memory dumps using CNN and traditional models. (As cited in [28]).
[32] Öztürk, A., & Hızal, S. (2024). Supervised classification of memory dumps using XGBoost. (As cited in [28]).
[33] Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (CCS '17), pp. 1285–1298.
[34] Guo, H., Yuan, S., & Wu, X. (2021). LogBERT: Log Anomaly Detection via BERT. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.
[35] Nedelkoski, S., et al. (2023). NeuralLog: A parsing-free approach to log analysis. (As cited in [29]).
[36] Yang, L., et al. (2023). LAnoBERT: System log anomaly detection based on BERT. (As cited in [29]).
[37] Elia, P. (2025). *Real-Time Automated Forensic Evidence Collection in Critical Systems: Leveraging Advanced Network Monitoring Tools for Enhanced Cybersecurity Incident Response* (Master's thesis). Politecnico di Torino.
[38] Compagnucci, A., et al. (2025). SAFARI: a Scalable Air-gapped Framework for Automated Ransomware Investigation. *arXiv preprint* arXiv:2504.07868. (Accepted at IFIP SEC 2025).
[39] Maltego Technologies GmbH. (2025). Maltego: Cyber investigation platform for OSINT and link analysis. [Software]. https://www.maltego.com/
[40] Carrier, B. (2025). The Sleuth Kit (TSK) and Autopsy. [Software]. https://www.sleuthkit.org/
[41] Canadian Institute for Cybersecurity. (2022). CIC MalMemAnalysis-2022 Dataset. https://www.unb.ca/cic/datasets/malmemanalysis-2022.html
[42] Hadoop Distributed File System (HDFS) Log Dataset. (As cited in [29]).
[43] Canadian Institute for Cybersecurity. (2017). CICIDS2017 Dataset. https://www.unb.ca/cic/datasets/ids-2017.html
[44] Moustafa, N., & Slay, J. (2015). UNSW-NB15 dataset. https://research.unsw.edu.au/projects/unsw-nb15-dataset