IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

CHRONOS: An AI-Powered Kernel Monitoring and Attack Visualization Platform Using eBPF Telemetry and Behavioural Graphs

Asuwath Kumaresh V[1], Jesvin Bruce J[2], P. Sai Mounisha[3]
Students, Department of Computer Science and Engineering (Cyber Security)[1,2]
Assistant Professor, Department of Computer Science and Engineering (Cyber Security)[3]
Sri Shakthi Institute of Engineering and Technology, Coimbatore, India

Abstract: *Security tools generate many endpoint alerts that analysts must correlate manually to understand an intrusion. This paper presents CHRONOS, an AI-assisted kernel-monitoring and attack-visualization platform that links low-level Linux telemetry to interpretable behavioural context. The design combines eBPF event collection, schema normalization, optional Kafka streaming, PostgreSQL/TimescaleDB storage, a time-aware NetworkX behavioural graph, rule- and model-assisted threat analysis, evidence-linked MITRE ATT&CK mapping, template-based narration and a React dashboard. A policy-gated response design keeps human approval before disruptive action. We describe the architecture, graph model, workflow, prototype components and a layered verification plan. CHRONOS is a research prototype; no accuracy, latency or overhead figures are claimed because reproducible measurements have not yet been collected. Its contribution is a traceability-first design in which every finding and narrative statement links to source events..*

Keywords: eBPF, kernel monitoring, behavioural graph, provenance graph, intrusion detection, explainable AI, MITRE ATT&CK, security visualization.

I. INTRODUCTION

Modern Linux hosts produce a continuous stream of process activity, file operations, network connections and service interactions. Security products can flag suspicious items in these streams, but the findings are usually presented as separate records. An analyst must then work out whether several records belong to one activity, which process initiated it, which resources it touched, and whether the sequence resembles a known attack technique. This manual correlation increases investigation time and makes the reasoning behind a conclusion difficult to audit.

Kernel-level observation is a useful source of context because most security-relevant actions eventually pass through operating-system interfaces. The extended Berkeley Packet Filter (eBPF) lets verified programs attach to selected kernel events and forward observations to user space, and it has been applied to runtime monitoring, supply-chain provenance, rootkit detection and forensics [1]–[4]. Raw telemetry, however, still needs normalization, storage, correlation and interpretation before it helps an analyst. Graph models preserve the relationships among events, so that a process can be connected to the file it modified, the socket it opened and the child processes it created.

This paper presents CHRONOS, an AI-assisted kernel-monitoring and attack-visualization platform built around these ideas. Its investigation lifecycle is Observe → Normalize → Correlate → Analyze → Explain → Review → Respond, and its central output is an investigation context: a timeline, a graph of related entities, ATT&CK technique mappings, risk indicators and a narrative tied to observed evidence. The contributions of this work are:

(i) a layered, modular architecture that separates collection, transport, storage, graph correlation, threat analysis and visualization so that each can be developed and tested independently; (ii) a time-aware behavioural graph model that keeps source-event references on every node and edge and distinguishes observed from inferred relationships; (iii) an

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-40338

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, October 2026 Impact Factor: 8.2

evidence-linked analysis path combining transparent rules, a baseline machine-learning model, ATT&CK mapping and deterministic narrative generation; (iv) a policy-gated response design that places human approval between detection and disruptive action; and (v) a reproducible verification plan that avoids unmeasured performance claims.

CHRONOS should be read as a research and demonstration codebase, not as a drop-in commercial endpoint detection and response product. The presence of a schema, interface or design document does not establish that an end-to-end production integration is deployed. Where this paper describes a capability, it states whether it is implemented in the prototype or remains a design target.

II. METHODOLOGY

A. System Architecture Overview

CHRONOS follows a layered design in which every derived graph relationship or finding retains references to the observations that produced it. Five layers are defined. The monitoring layer obtains telemetry from the operating system using eBPF probes. The streaming layer buffers and distributes records; FastAPI [5] provides HTTP ingestion and Apache Kafka [6] is an optional event bus, with a simpler direct-ingestion path for small deployments. The data layer uses PostgreSQL with TimescaleDB [7] for structured records and time-series summaries. The application layer validates events, builds the graph with NetworkX [8], and runs threat analysis. The presentation layer is a React [9] dashboard with Cytoscape.js [10] for graph exploration. Fig. 1 shows the end-to-end data flow from kernel probes to the dashboard.

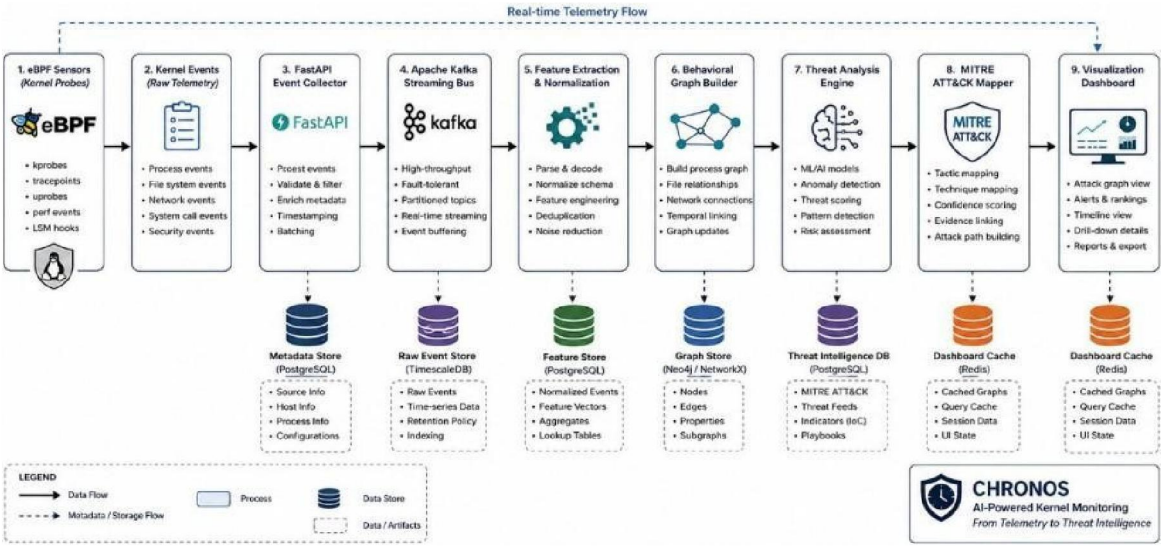


Fig. 1 End-to-end CHRONOS system architecture, from eBPF kernel sensors through FastAPI ingestion, Kafka streaming, feature extraction, behavioural graph construction, threat analysis and ATT&CK mapping to the visualization dashboard.

B. Design Principles

Six non-functional requirements shape the design. Traceability: an analyst can relate any narrative or recommendation to underlying events. Safety: detection does not automatically imply a disruptive response. Portability: CO-RE reduces kernel-version coupling, but compatibility still depends on kernel features, BTF availability, privileges and probe support. Resilience: event loss, malformed records and backpressure are handled explicitly. Privacy: command lines, paths and process metadata can be sensitive, so collection and storage are minimized and access-controlled. Maintainability: analysis stages are separable so that collectors, models, storage and UI evolve independently.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

C. Data Flow Contract

For an end-to-end deployment every event carries a stable event ID, timestamp, host identity, event type, process identity and relevant entity references. Derived objects retain source event IDs. Missing or uncertain attributes remain explicitly unknown rather than being inferred silently. This contract supports ordering, deduplication, graph construction, narrative citations and later replay. The eBPF sensor sources are CO-RE oriented and cover process creation and execution, networking, file operations, memory mapping or permission changes, and process tracing; a Go collector prototype forwards events, and topic and schema documentation describes the raw and enriched event classes.

D. Processing Workflow

A record passes through seven stages from collection to the analyst view, summarized in Table I. Stages can be exercised independently, and deployments may use only a subset.

TABLE I PROCESSING WORKFLOW STAGES

Stage	Processing	Output
1. Collection	Observe supported kernel and host events; attach host and time metadata	Raw event record
2. Validation	Check schema, timestamps, identifiers and required fields	Validated or rejected record
3. Normalization	Map event types and entity identifiers to a common schema	Normalized event
4. Correlation	Resolve entities; connect related actions over a time window	Graph nodes and edges
5. Analysis	Apply rules or a validated model to rank suspicious sequences	Finding with evidence
6. ATT&CK mapping	Associate supporting evidence with candidate techniques	Technique and rationale
7. Visualization	Render graph, timeline, filters and details	Analyst investigation view

E. Behavioural Graph Model

The graph is a directed, time-aware multigraph of entities and their observed interactions (Table II). Nodes have stable identifiers within the host or collection scope. Edges carry an action type, timestamp, source-event identifier and any available confidence or collection metadata. A multigraph is used so that repeated interactions between the same process and file or socket are retained. The graph does not imply causation where only temporal proximity is known; edge semantics reflect only what the sensor actually observed.

TABLE II: BEHAVIOURAL GRAPH ENTITIES AND RELATIONSHIPS

Entity / Relation	Example Fields or Meaning
Process node	Host ID, PID, parent PID, executable path, start time
File node	Path or stable file identifier, observed metadata, host ID
User node	User identifier or security principal, subject to data minimization
Socket / endpoint node	Protocol, local and remote endpoint, port, observation time
spawned edge	A process created another process; source event reference retained
read / write edge	A process interacted with a file; operation and time recorded
connected-to edge	A process or socket initiated or took part in a network connection
Finding annotation	Rule or model identifier, severity, rationale, supporting event IDs

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

F. Feature Extraction and Risk Propagation

The feature extractor derives process lineage and indicators such as suspicious memory permissions, sensitive-file activity and outbound targets. Feature definitions need explicit time windows and missing-data behaviour before results can be compared across hosts or experiments. The graph engine adds bounded risk propagation and a highest-risk-path operation. These are prioritization heuristics: their outputs are not calibrated probabilities unless calibrated experimentally, and a risk score never proves malicious intent, since debuggers use process tracing, runtimes request executable memory and administration tools modify sensitive files.

G. Threat Analysis and ATT&CK Mapping

Threat analysis begins with transparent rule-based patterns and can evolve toward supervised models when suitable data exists. A signature-based mapper associates behaviours with ATT&CK techniques, with current examples including command execution, ingress tool transfer, process injection and credential access. Each mapping records the technique identifier, a readable name, rule strength, matching evidence IDs and known limitations, and is presented as an evidence-supported hypothesis, not proof of compromise. ATT&CK is a behaviour taxonomy, not a severity scale, so severity is assessed separately. An intent-prediction package provides a baseline gradient-boosted model that ranks likely objectives such as credential access or data exfiltration; it is research scaffolding and, together with its synthetic-data utilities, does not establish real-world accuracy.

H. Evidence-Grounded Explanation

The explainability module separates extraction of an evidence bundle (graph path, risk, technique matches and source event IDs) from narrative rendering. Template-based narration is deterministic and independent of an external language model. An optional LLM path would receive only minimized, redacted evidence and is not treated as an independent source of facts. “Hallucination-free” narration is therefore an architectural goal for constrained, evidence-led text, not a guarantee; unsupported claims should be omitted or marked as inference. The timeline component orders evidence chronologically while exposing ambiguity from timestamp precision, buffering or late arrival.

I. Policy-Gated Response

A response engine separates policy evaluation from execution. Candidate actions include process isolation, network isolation, file quarantine and forensic capture. Disruptive actions are recommended for human approval, while narrowly defined non-destructive actions may run under policy. The executor validates targets, permissions, approval state and audit context, and each response record stores the triggering evidence, requested action, policy decision, approver, result, timestamp and rollback procedure. A score or model output alone must never authorize a destructive operation.

J. Implementation and Technology Stack

The prototype combines C/eBPF, Go, Python, SQL and TypeScript. Python implements analysis, graph operations, the API and tooling; C holds the eBPF programs; Go holds the collector prototype; SQL defines relational and time-series storage; and React with TypeScript provides the browser interface. Table III lists the technology chosen for each layer, with PyTorch [11] reserved for future graph-model experiments and Docker [12] used for repeatable packaging.

TABLE III: TECHNOLOGY STACK BY ARCHITECTURAL LAYER

Layer	Technology	Role in the Design
Kernel monitoring	eBPF, BCC, libbpf	Observe selected runtime events
Backend services	Python, FastAPI	Validation, ingestion, query endpoints
Event streaming	Apache Kafka (optional)	Buffering and distribution of telemetry
Persistence	PostgreSQL, TimescaleDB	Structured records, time-series summaries
Graph engine	NetworkX; Neo4j as scale option	Entity relationships and traversal

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

Analysis	scikit-learn, PyTorch	Baseline classifier; future model work
Frontend	React, TypeScript, Cytoscape.js	Interactive dashboard and graph view
Deployment	Docker	Repeatable packaging

The backend is a FastAPI application with modules for authentication, alerts, graphs, predictions and replay, Pydantic schemas for request and response structures, and ORM-based persistence. Proposed endpoints include health, event submission and query, filtered graph retrieval, findings with supporting evidence, ATT&CK mappings and analyst feedback. The dashboard combines summary metrics, an attack graph, an event timeline, technique matches, risk heatmaps and response review, and distinguishes live, simulated and seeded data so that demo state is not mistaken for live telemetry.

Table IV gives an honest account of implementation status. Items marked as prototype or contract material still require host-specific validation or deployment before any operational claim is made.

TABLE IV COMPONENT IMPLEMENTATION STATUS

Component	Current Role	Status Interpretation
eBPF sensors, collector	Event sources, definitions, collection	Prototype; host and kernel validation required
Streaming and database	Topic and event schemas, SQL schema	Contract material; deployment needed
Feature extraction	Behavioural feature derivation	Implemented; input quality affects results
Graph engine	NetworkX graph, risk utilities	Implemented local graph analysis
Intent prediction	Baseline model, training tools	Baseline research component; no accuracy claim
ATT&CK mapping	Behaviour-to-technique signatures	Implemented rule-based; coverage limited
Explainability	Evidence bundle, timeline, narrative	Implemented template path; LLM path unverified
Backend and frontend	FastAPI services; React dashboard	Implemented; deployment configuration needed
Response engine	Policy and action logic	Privileged execution needs isolated testing

K. Security and Privacy Controls

Kernel monitoring may need elevated capabilities, which should be confined to the collector and not inherited by analysis or UI processes. Services run with least privilege, transport uses TLS where appropriate, the dashboard requires authentication in shared environments, and secrets are loaded from environment variables or a secret store, never from source code. Telemetry can expose usernames, paths, command arguments and network endpoints, so collection is limited to the security purpose, stored records are role-controlled, retention matches the evaluation need, and test data is synthetic or sanitized. Authentication, input validation, audit logging and dependency management are treated as part of the system rather than optional additions.

III. LITERATURE REVIEW

A. Kernel Telemetry and eBPF

eBPF provides a controlled execution environment inside the Linux kernel. The BPF Compiler Collection (BCC) and libbpf offer different development and deployment paths, and CO-RE (Compile Once, Run Everywhere) with BPF Type Format (BTF) aims to reduce coupling to specific kernel versions. Published work uses eBPF for dependency

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

identification in software supply chains [1], policy enforcement in containerized environments [2], hidden-rootkit detection [3] and real-time forensic collection [4]. Open-source runtime-security projects such as Falco, Tetragon, Tracee and KubeArmor demonstrate mature eBPF-based observability and enforcement [13]–[16]. A practical collector must choose events carefully: excessive instrumentation adds overhead and volume, whereas narrow coverage can omit the evidence needed for correlation.

B. Provenance and Behavioural Graphs

Provenance analysis models how system entities interact over time. Nodes represent processes, files, users and sockets; edges represent actions such as spawn, read, write or connect. A survey of provenance-based intrusion detection summarizes the area [17], and recent work applies graph representation learning, deep learning and spatial-temporal or transformer-based models to detect advanced persistent threats and reconstruct attack scenarios [18]–[23]. These studies show the value of modelling context instead of classifying each event independently. Graph size is a recurring engineering concern: dense graphs are expensive to analyse and hard to display, so any reduction must preserve enough provenance to return to the underlying records.

C. Machine Learning and Explainability

Machine-learning models can rank suspicious sequences, but their output depends on training data, feature design and thresholds. For security operations a score alone is insufficient; analysts need to know which events and relationships contributed to a result. Counterfactual attack explanations for provenance-based detection [24] and surveys on explainable AI for threat intelligence and alert classification [25], [26] point in this direction. Explanations must clearly separate model inference from directly observed facts.

D. ATT&CK Alignment and Research Gap

MITRE ATT&CK provides a common vocabulary for adversary tactics and techniques [27]. Recent work uses machine learning to predict and map techniques and to estimate attack stage from temporal provenance data [28], [29]. The gap that CHRONOS addresses is not the absence of individual monitoring or detection tools but the need to combine telemetry, event relationships, explainable prioritization, technique mapping and investigation-oriented visualization in one traceable workflow. This is an integration objective, and its effectiveness must be established by implementation and repeatable testing rather than inferred from the architecture.

IV. RESULTS AND DISCUSSION

This section reports the current status of the work. CHRONOS is a prototype whose components have been implemented and exercised individually, but end-to-end measurements of detection accuracy, latency, throughput and resource overhead have not yet been collected. To avoid unsupported claims, this section therefore presents an illustrative investigation scenario, the verification plan that will generate measured results, and a discussion of limitations.

A. Illustrative Investigation Scenario

Fig. 2 shows the kind of behavioural attack graph the dashboard is intended to present. In this illustrative scenario, a browser process downloads a file, the file launches a shell process, the shell connects to an external address, and a secondary relationship shows access to a sensitive process. Each node and edge would link back to source event IDs, the side panel would list the candidate ATT&CK techniques with their supporting evidence, and the risk value would be a prioritization aid for the analyst. The figure is a design illustration; it is not output from a measured experiment.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

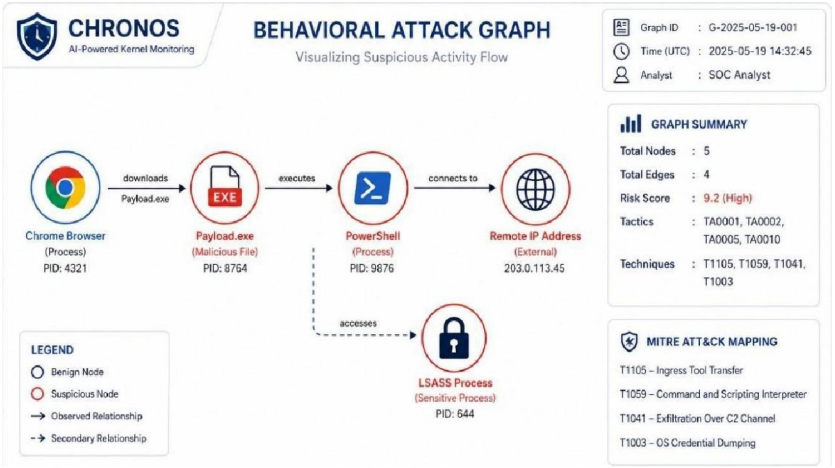


Fig. 2 Illustrative behavioural attack graph with node legend, graph summary and candidate ATT&CK mappings (design illustration, not measured output).

B. Verification and Evaluation Plan

Verification proceeds in layers and then through end-to-end scenarios. Unit tests cover event parsing, identifier handling, graph-edge creation, rule evaluation and serialization. Integration tests check that the API, storage, graph engine and dashboard exchange consistent records. End-to-end tests use controlled benign and simulated suspicious sequences in synthetic fixtures or an isolated lab. Table V lists the functional scenarios.

TABLE V FUNCTIONAL VERIFICATION PLAN

ID	Scenario	Expected Result
T01	Submit a valid normalized event	Accepted; event identifier returned
T02	Submit an event missing a required field	Rejected or quarantined with reason
T03	Send events for a process and child process	Expected process relationship created
T04	Send a process–file interaction	Edge carries a source-event reference
T05	Query graph by host and time range	Only matching entities returned
T06	Trigger a configured test rule	Finding has rule ID and event IDs
T07	Open a finding in the dashboard	Severity, rationale, evidence, graph visible
T08	Review a candidate ATT&CK mapping	Technique and supporting observations shown
T09	Disable or fail the collector	Health state shows telemetry unavailable
T10	Restart after events are stored	Persisted records remain queryable

Performance evaluation will record ingestion throughput, end-to-end latency, graph-query latency, CPU and memory use, storage growth, and dropped or rejected events, using percentile rather than only average latency and documenting hardware, kernel version, configuration and dataset. Detection quality will be reported as precision, recall and F1 on labelled cases, together with false positives per host and time period. The stage-wise plan also covers sensor coverage and overhead across a declared kernel matrix, normalization correctness against ground truth, graph entity-resolution quality, intent-model evaluation on held-out hosts or campaigns with calibration and simple baselines, explanation fidelity reviewed by analysts, response-safety tests in isolation (approval enforcement, invalid targets, replayed approvals, rollback), and analyst usability measures such as time to understand a scenario.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

C. Discussion and Limitations

The principal benefit CHRONOS aims for is investigative context rather than a claim that a particular algorithm will detect all attacks. Graph visualization can make relationships easier to inspect, but it introduces risk if inferred links are displayed as facts or if the display omits important events, so a sound evaluation must test both technical correctness and analyst usefulness.

Several limitations apply. Telemetry coverage depends on kernel version, permissions, probe availability and collector configuration, and missing telemetry must not be read as absence of activity. High-volume events can strain storage and processing and may require filtering or aggregation. Temporal association can be mistaken for causality. Machine-learning performance depends on the quality and representativeness of labelled data. Rule-based ATT&CK mappings cover a limited set of behaviours, can produce false positives and cannot represent every technique. Natural-language output can mislead if not validated against its evidence bundle, and risk propagation can amplify noisy input. Finally, the presence of modules does not prove a resilient end-to-end system: live ingestion, persistence, authentication, scale, fault tolerance and response execution each need independent verification.

V. CONCLUSION

This paper presented CHRONOS, an AI-assisted kernel-monitoring and attack-visualization platform that connects eBPF-oriented telemetry with event normalization, behavioural graph construction, threat analysis, ATT&CK mapping, evidence-grounded explanation and an interactive dashboard. Its strongest design principle is the linkage between low-level observations and analyst-facing conclusions, and its response design keeps human approval between detection and disruptive action. The work is a foundation for implementation and evaluation, not evidence of guaranteed threat detection.

Future work will (i) implement and benchmark a minimal end-to-end pipeline on a documented, reproducible dataset; (ii) add richer event lineage and explicit tracking of collector coverage and dropped events; (iii) evaluate graph-reduction strategies that preserve provenance; (iv) compare rule-based baselines with validated machine-learning models using precision, recall and F1, exploring temporal graph models only when enough labelled data exists; (v) enforce evidence-ID provenance for every generated narrative claim and test its faithfulness; (vi) strengthen deployment controls such as role-based access, TLS and automated security testing; and (vii) follow a staged deployment from local demonstration, through a single-host sensor prototype and a durable data path, to response simulation before any controlled response pilot.

ACKNOWLEDGMENTS

The authors thank Ms. P. Sai Mounisha , Assistant Professor and the faculty of the Department of Computer Science and Engineering (Cyber Security), Sri Shakthi Institute of Engineering and Technology, Coimbatore, for their guidance and technical suggestions, and the maintainers of the open-source technologies cited in this paper.

REFERENCES

01. N. Srinivasan, N. Naveen, and N. Naveen, “Bomfather: An eBPF-based kernel-level monitoring framework for accurate identification of unknown, unused, and dynamically loaded dependencies in modern software supply chains,” arXiv:2503.02097, 2025.
02. “eBPF-PATROL: Protective agent for threat recognition and overreach limitation using extended Berkeley Packet Filter (eBPF) in containerized and virtualized environments,” arXiv:2511.18155, 2025.
03. “Kernel-level hidden rootkit detection based on eBPF,” ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/392894872
04. “Enhancing DFIR in orchestration environments: Real-time forensic framework with eBPF for Windows,” ScienceDirect, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666281725000629
05. FastAPI Documentation. [Online]. Available: https://fastapi.tiangolo.com

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, October 2026
Impact Factor: 8.2

06. Apache Kafka Documentation. [Online]. Available: https://kafka.apache.org/documentation
07. Timescale Documentation. [Online]. Available: https://docs.timescale.com
08. NetworkX Documentation. [Online]. Available: https://networkx.org
09. React Documentation. [Online]. Available: https://react.dev
10. Cytoscape.js Documentation. [Online]. Available: https://js.cytoscape.org
11. PyTorch Documentation. [Online]. Available: https://pytorch.org/docs
12. Docker Documentation. [Online]. Available: https://docs.docker.com
13. Falco, "Cloud-native security tool designed for Linux systems." [Online]. Available: https://falco.org
14. Tetragon, "eBPF-based security observability and runtime enforcement." [Online]. Available: https://tetragon.io
15. Tracee, "Runtime security and forensics using eBPF." [Online]. Available: https://aquasecurity.github.io/tracee/latest
16. KubeArmor, "Runtime Kubernetes security engine." [Online]. Available: https://kubearmor.io
17. "Provenance-based intrusion detection systems: A survey," ACM Comput. Surv., 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3539605
18. "Provenance-based APT campaigns detection via masked graph representation learning," ScienceDirect, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167404824004644
19. "Provenance graph-based deep learning framework for APT detection in edge computing," Appl. Sci., vol. 15, no. 16, 2025.
20. "CONTINUUM: Detecting APT attacks through spatial-temporal graph neural networks," arXiv:2501.02981, 2025.
21. "TBDetector: Transformer-based detector for advanced persistent threats with provenance graph," arXiv:2304.02838, 2023.
22. "MGDA: A provenance graph-based framework for threat detection and attack scenario reconstruction," ScienceDirect, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128625007728
23. "Provenance-based intrusion detection via multi-scale graph representation learning," Springer Nature Link, 2025.
24. "ProvX: Generating counterfactual-driven attack explanations for provenance-based detection," arXiv:2508.06073, 2025.
25. "Survey perspective: The role of explainable AI in threat intelligence," arXiv:2503.02065, 2025.
26. "Evaluating explainable AI for deep learning-based network intrusion detection system alert classification," arXiv:2506.07882, 2025.
27. MITRE Corporation, "MITRE ATT&CK Enterprise Matrix." [Online]. Available: https://attack.mitre.org
28. "KillChainGraph: ML framework for predicting and mapping ATT&CK techniques," arXiv:2508.18230, 2025.
29. "Learning the APT kill chain: Temporal reasoning over provenance data for attack stage estimation," arXiv:2603.07560, 2026.