SysArmor: The Practice of Integrating Provenance Analysis into Endpoint Detection and Response Systems

Shaofei Li*, Jiandong Jin*, Hanlin Jiang*, Yi Huang*, Yifei Bao†, Yuhan Meng*
Fengwei Hong*, Zheng Huang*, Peng Jiang‡, and Ding Li*
*Peking University, †Jilin University, ‡Southeast University
lishaofei@pku.edu.cn, jjd@pku.edu.cn, jianghanlin@stu.pku.edu.cn, yihuang@stu.pku.edu.cn
baoyf2122@mails.jlu.edu.cn, mengyuhan@pku.edu.cn, fwhong@stu.pku.edu.cn, zhenghuang24@stu.pku.edu.cn
pengjiang@seu.edu.cn, ding_li@pku.edu.cn

***Abstract*—Endpoint Detection and Response (EDR) systems play a crucial role in modern cybersecurity by monitoring and responding to Advanced Persistent Threats (APT) on endpoints. Provenance analysis has emerged as a powerful technique for enhancing EDR capabilities by providing detailed insights into system activities and enabling advanced threat detection. However, enterprises still face significant challenges in effectively processing and analyzing provenance data for real-time threat detection and response. In this paper, we present SysArmor, a practice of integrating provenance analysis into EDR systems designed to address these challenges through a novel microservices architecture. SysArmor integrates efficient provenance data collection, real-time streaming processing, and asynchronous detection engine that combines Falco rule-based detection with provenance graph-based anomaly detection, NodLink and KnowHow, to provide end-to-end online threat detection. To help security analysts investigate alerts, SysArmor offers a management front end that manages alerts and visualizes provenance graphs. We deploy SysArmor in a real-world enterprise environment and evaluate its performance and effectiveness. Our results demonstrate that SysArmor can detect real-world APT attacks effectively while maintaining high throughput and low latency. SysArmor is also scalable and can be easily deployed in multiple endpoints.**

I. Introduction

Endpoint Detection and Response (EDR) systems have become an essential component in modern cybersecurity strategies, providing organizations with the ability to detect, investigate, and respond to Advanced Persistent Threats (APT) on endpoints. Companies such as Palo Alto Networks [1], CrowdStrike [2] and Wiz [3] have popularized EDR solutions that offer real-time monitoring, threat detection, and incident response capabilities. Despite their widespread adoption, existing EDR systems face a critical limitation that hinder their ability to combat modern cyber threats effectively: *they struggle to adapt to rapidly evolving threats, lack the precision needed for accurate detection, and fail to offer comprehensive protection against sophisticated cyberattacks*. First, traditional EDR solutions primarily rely on Security Information and Event Management (SIEM), which often fall short in identifying sophisticated attacks that leverage novel techniques or exploit zero-day vulnerabilities [4]. Second, inadequate behavioral analysis results in unsophisticated models that trigger false positives by misclassifying normal user behaviors while failing to detect subtle, sophisticated abnormal behaviors of advanced attacks [4], [5]. Thirdly, limited traceability and forensics capabilities mean these systems can only offer partial information during security incidents, preventing security teams from tracing the complete attack path, identifying root causes, and generating the interpretable alerts necessary for effective incident response [6].

To address these limitations, recent advancements in system provenance analysis have been investigated as a promising approach for enhancing EDR capabilities [7]–[13]. System provenance captures detailed records of system activities, enabling the reconstruction of attack paths and the identification of malicious behaviors that may evade traditional detection methods. Recent researches have proposed various Machine Learning (ML)-based techniques to analyze provenance data for anomaly detection and threat hunting. These techniques leverage graph-based representations and advanced tracking algorithms to improve detection capabilities and provide richer contextual information for incident investigation.

However, according to the survey of the gap between academic research and commercial deployment of EDR systems [6], provenance analysis is still underutilized due to several challenges. First, ***efficient collection of provenance data.*** Collecting comprehensive system provenance data at scale without significant performance overhead remains a major challenge. Second, ***real-time processing and analysis of large-scale provenance data.*** Processing massive volumes of provenance data in real-time requires sophisticated streaming architectures and efficient ML-based detection algorithms. Third, ***alert interpretation and visualization.*** Security analysts face difficulties in interpreting and visualizing complex

Workshop on Attack Provenance, Reasoning, and Investigation for Security in the Monitored Environment (PRISM) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-00-8
https://dx.doi.org/10.14722/prism.2026.23002
www.ndss-symposium.org

provenance-based alerts and understanding the complete attack context. These challenges have hindered the widespread adoption of provenance-based EDR systems in practice.

To investigate and bridge this gap, we propose a framework, SYSARMOR, that integrates advanced provenance analysis techniques into existing EDR systems. SYSARMOR addresses these challenges through a novel microservices architecture and advanced threat detection capabilities. SYSARMOR introduces several key innovations:

- **Real-time Stream Processing**: Leveraging Apache Flink to enable high-throughput, low-latency processing of provenance data streams from multiple endpoints.
- **Asynchronous Hybrid Detection with Rule and ML Engines**: A multi-layer detection pipeline that integrates rule-based analysis with machine learning-based inference to support scalable and flexible threat detection.
- **Alert Interpretation and Visualization**: Intuitive dashboards and visualization tools that facilitate alert interpretation and support efficient incident response.

In our practice of integrating provenance analysis into EDR systems with SYSARMOR, we find that it is possible to overcome the challenges mentioned above and use latest provenance analysis techniques to enhance the capabilities of existing EDR solutions. Our system demonstrates promising improvements in detection capability, processing efficiency, and operational flexibility compared to traditional EDR solutions. It provides a scalable and effective approach to leveraging system provenance for enhanced endpoint security. Through comprehensive evaluation, we show that SYSARMOR effectively collects and analyzes provenance data for sophisticated attack patterns while maintaining low false positive rates and minimal performance impact on monitored systems.

During the deployment of SYSARMOR in the Third Research Institute of Ministry of Public Security, we identify several key challenges and lessons learned from real-world usage. These insights provide valuable guidance for future research and development in provenance-based EDR systems. The main contributions of this paper are as follows:

- The design of SYSARMOR, an end-to-end provenance-based EDR system with efficient data collection, real-time streaming, asynchronous hybrid detection with rule and ML engines, and alert interpretation and visualization.
- Empirical and real-world deployment demonstrating SYSARMOR's effectiveness and scalability in detecting APT attacks.
- Comprehensive analysis of challenges and lessons learned from deploying provenance-based EDR systems in practice.
- Open sourcing SYSARMOR to facilitate further research and development in provenance-based endpoint security at https://sysarmor.pku.edu.cn/.

II. BACKGROUND

A. EDR System Architectures

EDR systems represent an evolutionary advancement over traditional antivirus solutions [14], [15], addressing fundamental limitations in detecting APT attacks. EDR system provides continuous monitoring and comprehensive analysis of endpoint activities, enabling precise detection, investigation, and response to security incidents. Modern EDR architectures typically consist of two primary components: the EDR client and the EDR server. Figure 1 illustrates the high-level architecture of a typical EDR system.

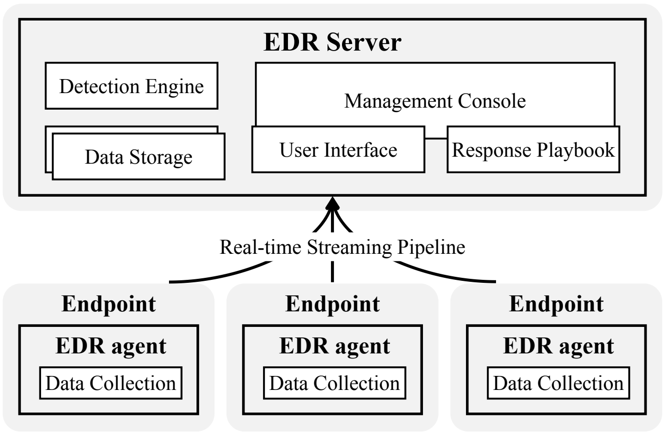

Fig. 1: High-level Architecture of Typical EDR Systems

EDR Client. EDR clients are software agents installed directly on monitored endpoints, responsible for comprehensive data collection and forwarding. These agents capture detailed system activity data including process execution patterns, file access operations, network connections, and registry modifications. Then the collected data are transmitted to the EDR server for centralized analysis. The clients need to guarantee low performance overhead to minimize impact on endpoint operations while ensuring reliable and secure data collection.
EDR Server. EDR server provides centralized processing, analysis, and management capabilities for security operations. Detection engines are deployed as the core analytical components that process data from multiple endpoints using techniques including rule-based detection and ML algorithms to identify potential threats. The server also provides management consoles for security analysts to monitor alerts, investigate incidents, and execute response actions, serving as central coordination points for security operations. Data storage systems maintain historical data for forensic analysis. The server must scale to handle data from multiple endpoints while providing real-time analysis capabilities for timely threat detection and response.

B. Endpoint Data Collection

Endpoint data collection forms the foundation of effective security monitoring by systematically capturing detailed information about system activities. It is always integrated into EDR client to provide the necessary visibility into endpoint behavior for threat detection and investigation. Modern operating systems provide sophisticated auditing framework that capture granular security-relevant events, enabling visibility into system operations while supporting security investigations and compliance requirements [16]–[18].

On Linux, the Audit Framework (auditd) [16] records security-relevant events—process execution (execve, fork, exit), file I/O (open, read, write, close, unlink), sockets and network connections, authentication, privilege changes, and system call parameters—using rule-driven, fine-grained filtering. On Windows, Event Tracing for Windows (ETW) [17] offers similar coverage for process, file, registry, and network activity via efficient kernel instrumentation and selective event filtering. Together, these facilities provide baseline endpoint visibility for detecting unauthorized or suspicious behavior.

C. Provenance Analysis

Provenance analysis represents a fundamental advancement in security monitoring by focusing on the complete historical record of system activities and their intricate causal relationships, providing a comprehensive view of system behavior over time. This approach enables sophisticated reconstruction of attack paths by systematically tracing causal dependencies between system events, allowing security teams to understand not just what happened during security incidents, but how and why specific events occurred.

Provenance Graph Representation. Provenance data is typically represented as directed graphs that capture complex relationships between system entities and their interactions [19]. In this representation, nodes correspond to system entities including processes, files, and network sockets, each representing distinct components of the observed system. Edges between these nodes represent causal relationships between entities, such as processes creating files, processes reading files, or processes connecting to network resources.

Provenance-based Detection. Provenance-based detection approaches leverage rich contextual information embedded within provenance graphs to identify sophisticated attack patterns that frequently evade traditional detection methods [8]–[11], [20]–[22]. By analyzing the structural and contextual properties of provenance graphs, such as subgraph patterns, connectivity metrics, temporal sequences, and event attributes, these approaches can identify anomalous behaviors indicative of APT. Machine learning techniques, including graph neural networks and unsupervised anomaly detection algorithms, are often employed to learn normal behavior patterns from historical provenance data and identify deviations that may signal malicious activities. Due to the complexity and scale of provenance data, efficient algorithms and scalable processing frameworks are essential to enable real-time analysis and detection capabilities in practical deployments [6], [23].

Alert Interpretation and Visualization. Provenance-based detection techniques generate alerts enriched with contextual information derived from provenance graphs, enhancing the interpretability and actionability of security alerts for analysts [6], [23]. Recent research has proposed various summarization [24]–[26] and interpretation [13], [27]–[30] techniques that distill complex provenance data into concise, human-readable formats, enabling analysts to quickly grasp the essence of detected threats. They leverage Cyber Threat Intelligence (CTI) reports and attack pattern recognition to provide meaningful explanations for alerts, facilitating effective incident investigation and response. Visualization techniques are also needed to present provenance data in reasonable arrangements and interactive interfaces, allowing analysts to explore attack paths, relationships between entities, and temporal sequences of events interactively.

III. Design

A. System Architecture Overview

SysArmor employs a microservices architecture that strategically separates concerns across four distinct layers: data collection, threat detection engine, storage, and management. This modular design enables stream processing of massive volumes of provenance data, providing exceptional scalability, robust fault tolerance, and enhanced maintainability while supporting real-time threat detection at enterprise scale.

The data collection layer forms the foundation of SysArmor's monitoring capabilities. It utilizes existing Linux auditing infrastructure through Linux Audit Framework (auditd) [31] and rsyslog [32], which significantly reduces deployment complexity and performance overhead while maintaining comprehensive system visibility. The Vector [33] data router efficiently collects and routes audit data to Kafka [34] message brokers, which provide durable, scalable event streaming with strong ordering guarantees essential for maintaining event causality in provenance analysis.

At the core of the processing pipeline lies the threat detection engine, which utilizes Apache Flink [35] for distributed real-time threat detection. Event normalization components standardize diverse audit event formats into consistent processing structures. The threat detection component consists of two modulars: a rule-based detection engine and a machine learning inference engine. The rule-based engine implements a Falco-compatible [36] rule engine with advanced optimization features. ML inference interface integrates state-of-the-art ML-based detection algorithms, NodLink [9] and KnowHow [13], to identify APT attacks and interpret them with Tactics, Techniques, and Procedures (TTPs) techniques.

The storage layer employs a dual-database strategy that optimizes for different data access patterns. OpenSearch [37] serves as the primary storage for triggered alerts, providing powerful full-text search capabilities and efficient time-series data handling essential for attack investigations. PostgreSQL [38] manages system metadata and configuration data, leveraging its strong consistency guarantees and relational capabilities for operational data management.

The management layer provides centralized control and visibility through a RESTful Manager API [39] that exposes core system functionality and enables integration with external platforms. A modern React-based web interface [40] offers intuitive alert visualization and investigation workflows for security analysts, supporting efficient investigation and response. Additionally, SysArmor supports seamless interoperability with existing security infrastructure, including native integration with Wazuh for coordinated detection, correlation, and management.

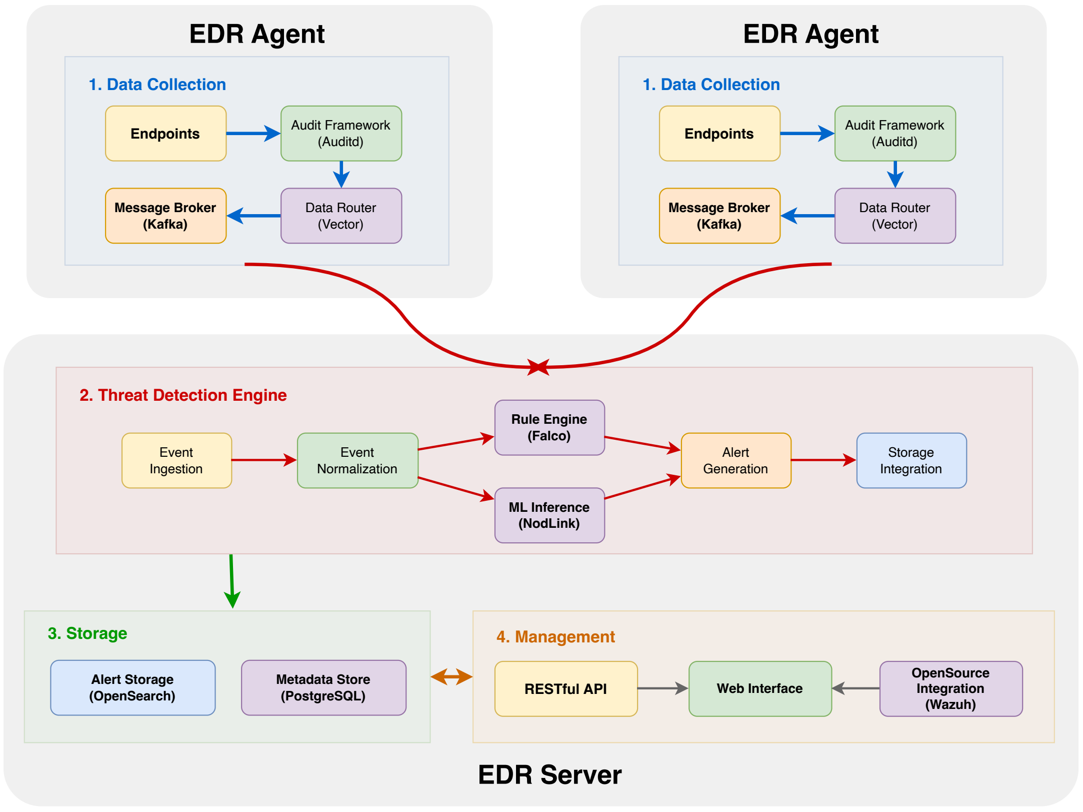

Fig. 2: SYSARMOR System Architecture

B. Data Collection

SYSARMOR's leverage existing audit infrastructure in common operating system, achieving provenance data collection without requiring additional kernal modules or agents. This approach minimizes deployment complexity while maintaining rich system visibility essential for effective threat detection.

Audit Infrastructure Integration. SYSARMOR integrates with the auditd for Linux and ETW for Windows to capture granular system activities. The system employs configurable audit rules that precisely define which security-relevant events to monitor. This includes coverage of process execution and lifecycle events, file system operations, network connections, user authentication activities, and privilege changes. Performance optimization is achieved through selective monitoring strategies that balance security coverage with system overhead, ensuring minimal impact on production workloads. SYSARMOR can be easily extended to support other operating systems by intergating into existing Kafka-compatible audit collectors.

Rsyslog Forwarding. Audit events are efficiently forwarded to the central processing infrastructure using rsyslog, which provides reliable TCP transport for event delivery over network connections. The system implements queue management with buffering and retry mechanisms to handle network failures gracefully, ensuring no security events are lost during transient connectivity issues. Crucially, rsyslog configuration preserves the original audit event structure and metadata, maintaining the integrity of provenance information essential for accurate threat detection and investigation.

Vector Data Router. The Vector data router serves as a high-performance event collection and routing component capable of processing thousands of events per second while maintaining reliable delivery semantics. It implements exactly-once processing guarantees for critical security events, ensuring that no events are duplicated or lost during transmission. The router performs intelligent format conversion, transforming diverse audit event formats into standardized JSON structures that facilitate consistent processing throughout the detection pipeline. This standardization enables efficient parsing and analysis while preserving all relevant security context.

C. Real-time Stream Processing

SYSARMOR employs Apache Flink as the foundation for real-time processing of provenance data streams. To simplify configuration and achieve predictable scaling, SYSARMOR adopts a unified topic architecture based on Kafka native partitioning, refactored from dynamic multi-topic patterns to a small set of fixed topics with partition-based scaling. The Flink processing pipeline is implemented as a staged set of independent jobs that communicate via the unified Kafka topics, enabling independent scaling, versioned deployments, and isolated fault recovery for each stage.

Unified Topic Architecture and Three-layer Data Flow. The data plane uses a three-layer topic organization that enforces semantic separation while keeping the topic count stable (six fixed topics total across logical groups): *Raw Data layer* stores original audit records (32 partitions, 3-day retention). *Events layer* stores normalized events derived from raw data (32 partitions, 7-day retention). *Alerts layer* stores generated detection alerts (16 partitions, 30-day retention). This layered design reduces operational complexity, centralizes retention/partitioning policy, and avoids dynamic topic explosion. Kafka native partitioning (collector_id as the primary partition key) provides deterministic routing and load-balancing, enabling near-linear scaling to support 1000+ collectors in large deployments.

Flink Processing Pipeline. The Flink processing pipeline is implemented as a staged set of independent jobs that communicate via the unified Kafka topics. *Job 01: Event Normalization* consumes the Raw Data topic, performs event parsing and normalization, and writes normalized records to the Events topic. *Job 02: Rule-based Detection* consumes normalized Events, applies rule-based and ML-assisted detection logic, and produces structured alerts to the Alerts topic and OpenSearch. *Job 03: Asynchronous ML-based Inference* consumes Events topic to invoke external ML inference services asynchronously and write inference results back to Alerts for enrichment. Separating processing into three jobs enables independent scaling, versioned deployments, and isolated fault recovery for each stage.

Parallel Processing and Fault Tolerance. Horizontal scaling is achieved through Kafka partitioning and Flink parallelism. Using collector_id as the partition key ensures that events from the same endpoint are routed to the same partition and processing slot, preserving causal ordering required for provenance analysis while enabling balanced load distribution across workers. This design follows Kappa architecture principles (stream-first, single processing path) to avoid Lambda-style dual-path complexity. Flink checkpoints and Kafka transactional commits allow each job to recover independently with exactly-once guarantees, while the three-stage pipeline supports targeted autoscaling to meet workload spikes without global redeployment.

D. Threat Detection Engine

SysArmor employs an asynchronous decoupled dual-engine architecture that combines Falco rule-based detection [36] with provenance graph-based anomaly detection, NodLink [9] and KnowHow [13]. To prevent ML inference latency from blocking alert generation, the system adopts a fire-and-forget asynchronous communication pattern: the rule engine generates alerts while simultaneously sending normalized event data to the inference service in a completely decoupled manner. This hybrid architecture leverages the strengths of both detection methodologies, enabling the system to identify both known attack patterns through precise rule matching and emerging threats through adaptive machine learning analysis.

1) Rule-based Detection: SysArmor implements a compiler-based threat detection rule engine that combines Falco rule syntax with compiler optimization techniques. The compilation phase transforms Falco rules into efficient executable detection logic. Then the rule matching phase evaluates the optimized rules against incoming events using efficient pattern matching algorithms.

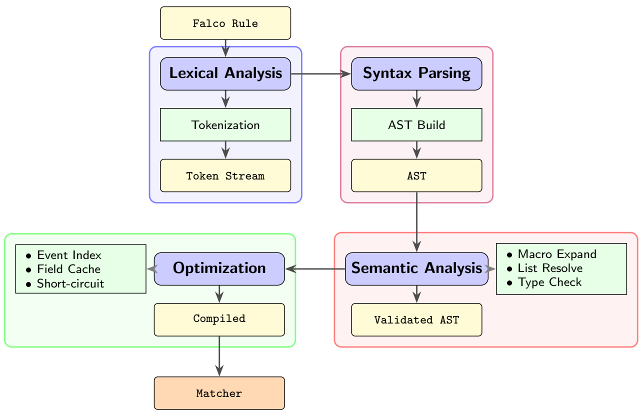

Fig. 3: Rule compilation pipeline with four phases

```
rule suspicious_tmp_execution {
  condition:
    (evt.type in (execve, execveat) and
    (proc.exe startswith "/tmp/" or
    proc.exe startswith "/dev/shm/"))
}
```

Fig. 4: Example Falco rule suspicious_tmp_execution for detecting process executions from temporary directories.

Compilation Phase. Figure 3 illustrates the rule compilation pipeline with four distinct phases. During compilation, Falco rules are converted into Abstract Syntax Tree (AST) that are suitable for high-throughput matching. The compilation phase comprises lexical analysis and AST construction coverting rule conditions into structured representations, semantic analysis expanding macros and resolves list references, and a set of optimizations that enhance runtime performance through event type indexing, field caching, and short-circuit mechanism.

Lexical analysis transforms the YAML-style rule into a sequence of atomic lexemes. *BinaryOpNode("and", In(FieldNode("evt.type"), SetNode({"execve","execveat"})), BinaryOpNode("or", StartsWith(FieldNode("proc.exe"), "/tmp/"), StartsWith(FieldNode("proc.exe"), "/dev/shm/")))* Then the semantic analysis phase validates the AST's correctness and resolves symbolic references. It expands macro definitions recursively, resolves list references to their concrete values, performs type checking to ensure field references match expected data types, and validates that all referenced fields exist in the event schema. This phase ensures that only well-formed, semantically valid rules proceed to the optimization stage, preventing runtime errors.

To optimize runtime performance, the compiler applies several advanced techniques. Event type indexing constructs an index mapping event types to relevant rules, enabling rapid pre-filtering of candidate rules during matching. Traditional rule engines evaluate all N rules against each incoming event, resulting in $O(N)$ complexity. SYSARMOR's compiler analyzes each rule's AST during compilation to extract event type constraints (e.g., 'evt.type = execve'). At runtime, when an event e arrives with event type t, the engine performs an $O(1)$ hash table lookup to retrieve only the subset of rules $R_t \subset R$ relevant to that event type, where $|R_t| \ll |R|$. Additionally, field caching through CachedEventData avoids redundant field parsing across multiple rule matchings with complexity $O(1)$ per cached field, and short-circuit mechanism terminates logical expressions (AND/OR) as soon as the outcome is determined.

Rule Matching Phase. The rule matching stage executes inside the streaming detection jobs and match incoming normalized events with compiled rules in real-time. At runtime, an incoming event is first used to lookup the event type index to produce a candidate rule set. Required fields are extracted on-demand and cached in the event context. Predicates are evaluated in a cost-ordered sequence and exploit short-circuit mechanism to avoid unnecessary computation. When a rule is satisfied, the system applies suppression and rate-control policies before constructing a structured alert (including rule identifier, matched fields, and provenance context) and emitting it to the Alerts Kafka topic and to OpenSearch for storage and subsequent analysis.

2) ML-based Detection: The ML-based detection component intergrates latest online provenance-based detection algorithms, NODLINK [9] and KNOWHOW [13], to identify APT attacks and interpret them with TTPs techniques.

NODLINK and KNOWHOW Integration. We choose NODLINK as the basic ML detection model due to its superior performance in online APT detection and unsupervised learning capability. It models APT detection on provenance graph as the Online Steiner Tree problem [41] and proposes an efficient approximation algorithm to solve it in real-time. NODLINK employs lightweight unsupervised Variational AutoEncoder (VAE) to identify suspicious terminals. This makes NODLINK well-suited for easy depolyment and timely updates in diverse enterprise environments. Additionally, we integrate KNOWHOW behind NODLINK to provide TTPs-based interpretation of detected attacks on the provenance graph. KNOWHOW leverages the General Indicator of Compromise (gIoC) extracted from CTI reports to identify TTPs techniques used in the attack campaign. This combination enables SYSARMOR to not only detect APT but also provide actionable insights into attacker behaviors and techniques.

Containerized Inference Service. To operationalize ML detection, SYSARMOR provides a containerized inference service that hosts NODLINK and KNOWHOW and exposes a stable HTTP API for asynchronous inference. The service is implemented as a lightweight FastAPI [42] web application that supports update training of the deployed models and serves inference requests from the Events topic. We implemented three types of API surface: training, inference, and management. Training API is used to update the model with newly collected data. Inference API supports batch events inference. Management API allows check the health status of the service and delete old models.

We then designed the inference service to be modular and extensible, allowing new detection algorithms to be added with minimal changes to the core architecture. Therefore, we dockerized the service to facilitate easy deployment and intergate it as *Job 03* in the Flink processing pipeline. A batch of normalized events are sent to the inference service asynchronously via HTTP POST requests, and the inference results are written back to the Alerts topic for enrichment. This design enables SYSARMOR to easily incorporate new ML-based detection algorithms in the future.

E. Storage and Query Infrastructure

SYSARMOR employs a dual-storage architecture that combines the complementary strengths of relational and search-oriented databases to achieve optimal performance and flexibility across different data access patterns. This architectural approach enables efficient handling of both structured operational data and unstructured security alert data while maintaining the specific advantages of each database technology.

Dual-Storage Architecture. OpenSearch is employed as the primary alert store, offering full-text search, aggregations, time-series optimizations and horizontal scalability to support rapid investigations and trend analysis; PostgreSQL stores operational and configuration data—collector registration and status, rule/versioning, global settings, and user authentication/authorization—providing strong consistency and relational semantics for management functions.

Query Optimization. The system employs multiple query optimization techniques that ensure responsive performance even under heavy query loads. Comprehensive indexing of frequently queried fields accelerates common search patterns and analytical queries, reducing query response times significantly. Query result caching stores frequently accessed query results, improving performance for repetitive queries and dashboard operations. Time-based partitioning of alert data organizes historical information into manageable segments, enabling efficient querying of specific time ranges without scanning entire datasets. Data compression techniques reduce storage requirements for historical alert data while maintaining query performance, enabling cost-effective long-term retention of security information.

F. Management and Integration

SYSARMOR provides management capabilities and extensive integration points designed to support enterprise security operations at scale. The management layer enables centralized control and visibility across distributed deployments while maintaining the flexibility required for diverse organizational environments and security workflows.

RESTful API. The Manager API exposes comprehensive functionality through a RESTful interface that enables programmatic control and integration with external systems. Collector management endpoints handle registration, status monitoring, and configuration of monitored endpoints, providing centralized control over the entire deployment. Alert querying capabilities support sophisticated search, filtering, and aggregation of security alerts, enabling automated processing and integration with security orchestration platforms. System monitoring endpoints provide health checks, performance metrics, and operational status information essential for maintaining system reliability and performance. Rule management functionality enables creation, modification, and deployment of detection rules, supporting dynamic threat response and security policy adaptation.

Web User Interface. The React-based web interface provides intuitive access to system functionality through a modern, responsive design optimized for security analyst workflows. The dashboard offers real-time overviews of security posture and recent alerts, enabling rapid assessment of organizational security status. Alert investigation tools provide detailed analysis capabilities and contextual exploration features that help security teams understand the complete scope and impact of detected threats. We also intergrate Large Language Model (LLM) to generate the human readable report of detected APT attacks. The analysts can interact with the LLM through a chat-style interface to help them better understand the attack details and suggest mitigation strategies.

Wazuh Integration. SYSARMOR includes integration with the Wazuh [43] security platform, enabling organizations to leverage existing Wazuh deployments while benefiting from SYSARMOR's advanced provenance-based detection capabilities. Agent management integration provides unified management of Wazuh agents and SYSARMOR collectors, simplifying operational administration across both platforms.

IV. EVALUATION

In this section, we introduce the performance of SYSARMOR through a real-world deployment in the Third Research Institute of Ministry of Public Security. We propose the following research questions to guide our evaluation:

- **RQ1: Detection Effectiveness.** What's the detection capability of SYSARMOR against APT attacks?
- **RQ2: Scalability.** How does SYSARMOR perform in terms of processing throughput? Can it scale effectively to large enterprise environments?
- **RQ3: Resource Consumption.** What is the memory consumption of SYSARMOR when processing real-time provenance data streams?

A. Evaluation Setup

We deploy SYSARMOR in the Third Research Institute of Ministry of Public Security for one week to evaluate its effectiveness and performance. The deployment encompasses the following configurations:

Test Environment. We deploy SYSARMOR agents on one endpoint within the Third Research Institute of Ministry of Public Security's network and monitor system activities over a period of one weeks. During the deployment, the red team at the Third Research Institute of Ministry of Public Security launched an APT attack campaign targeting the monitored endpoints to simulate real-world attack scenarios. The endpoint and server run Ubuntu on them. During the deployment, we collected approximately 30 million auditd events and convert them to 8,179,548 Sysdig events, which are then processed by SYSARMOR for real-time analysis. Therefore, the average event generation rate for per endpoint is around 57 events per second.

TABLE I: The detected rules by the rule-based detection engine of SYSARMOR during the real-world deployment.

Rule Name	Alert Count
command_injection_indicators	7600
crontab_modification	7525
permission_modification	6670
suspicious_binary_execution	4865
clear_log_activities	4454
dev_shm_execution	3677
temp_executable_creation	3673
nonstandard_port_connection	103
sudo_privilege_escalation	12
file_deletion_dangerous	5
service_manipulation	4

SYSARMOR Server Environment. We deploy the SYSARMOR server on a server with 32 CPU cores, a NVIDIA GeForce RTX 3090 GPU, 128 GB RAM, and two TB storage, running Ubuntu 20.04 LTS with Docker and Kubernetes for container orchestration. We use OpenSearch and PostgreSQL database for alert storage and metadata storage. We setup a website interface using React on the server for security analysts to monitor and manage the system.

B. RQ1: Detection Effectiveness

1) Rule-based Detection: We adapt existing rule sets from Falco [36], which contains over 36 rules for detecting various attack techniques and behaviors. During the deployment period, the rule-based detection engine of SYSARMOR successfully detected 38,588 alert events, about 0.47% of the total processed events. The results of detected rules are summarized in Table I. We can observe that the most frequently triggered rules are related to command injection indicators, crontab modification, permission modification, suspicious binary execution, log clearing activities, and execution from shared memory or temporary directory. These rules correspond to common techniques used by attackers to gain persistence, escalate privileges, and evade detection.

2) ML-based Detection: During the deployment period, ML-based detection engine of SYSARMOR, NODLINK + KNOWHOW successfully detected one APT attack attempts. This attack lasted for three days and the engine started to raise alerts on the first day of the attack, which shows the response capability of SYSARMOR against real-world APT attacks. The alert continued to evolve during the entire attack

TABLE II: The detection results of NODLINK + KNOWHOW inference engine of SYSARMOR against the real-world APT attack. NODLINK generates the provenance graph of the detected attack activities, and KNOWHOW maps the detected malicious events to MITRE ATT&CK TTPs and attack stages.

Attack Stage	TTPs	Malicious Command Line
Initail Compromise&	T1570	*/tmp/.>_, /var/tmp/.>_, /tmp/sshd, /root/sshd*
Establish Foothold &	T1105	*/usr/sbin/sshd -D, sshd [accepted], sshd nobody [priv]*
Move Laterally	T1021.004	*sshd root [priv], sshd operator [priv]*
	T1222	*(cd '/var/tmp' && [-f 'yum.log'] && (chattr -i 'yum.log'; chmod +x 'yum.log' && ./'yum.log'))*
Escalate Privilege	T1036	*(cd '/var/tmp' && [-f 'apt.log'] && (chattr -i 'apt.log'; chmod +x 'apt.log' && ./'apt.log'))*
	T1022.002	*chmod +x yum.log, chmod +x .bash_logout, /usr/bin/ydvlmeowey su 12812*
	T1016	*/usr/bin/yzhulutzjr ls -la, /usr/bin/ukkoisjdsi route -n, /usr/bin/dcjwwfaznr ifconfig eth0*
Internal Reconnaissance	T1057	*/usr/bin/zqjxlypgjo netstat -an, /usr/bin/rvwcaoweok ps -ef, /usr/bin/tseyeejrpq who*
	T1033	*/usr/bin/wrrvfqlxbh whoami, /usr/bin/mbydvjvtfs top, /usr/bin/srtewbasqq uptime, /usr/bin/cqlmxscjbm pwd*
	T1083	*/usr/bin/bkzytqxfba cd /etc, /usr/bin/vyqlcyypoo cat /etc/resolv.conf, which apt, grep -v grep*
	T1562	*shell -c ps -eo pid,comm,%cpu | awk '...' | sort -k3nr | ... | xargs kill ... || pgrep -x '>_' — tail*
Maintain Persistence	T1499	*readlink -f /proc/10751/exe, xargs -r kill -9, ps -o pid=,pcpu= -p 19901 30569*
	T1053	*chattr -i .bash_logout, /bin/bashell -c run-parts /etc/cron.hourly, basename /etc/cron.hourly/gcc.sh*

period, providing continuous monitoring and detection of the attack activities. When the attack concluded on the third day, the alerts ML-based detection engine did not update further, indicating that the attack had ceased. Finally, NODLINK + KNOWHOW generated an alert containing a provenance graph with 201 nodes and 1,982 edges.

Table II illustrates the core detection results, including the attack stages, TTPs, and malicious command lines. We removed the replicated or similar command lines for better readability. We can observe that SYSARMOR effectively detected six attack stages, including Initial Compromise, Establish Foothold, Move Laterally, Escalate Privilege, Internal Reconnaissance, and Maintain Persistence. During the three-day attack period, we did not observe Data Exfiltration or Impact stages, indicating that the attacker did not proceed to these phases. For each attack stage, SYSARMOR successfully mapped the detected malicious events to corresponding MITRE ATT&CK TTPs [44] and provided detailed command lines used by the attacker. We can observe that the attacker employed various techniques, such as T1570 (Lateral Tool Transfer), T1105 (Ingress Tool Transfer), T1036 (Masquerading), and T1562 (Impair Defenses).

Crucially, these TTP-level detection results are enabled by KNOWHOW 's semantic reasoning over structured cyber threat intelligence, specifically gIoC. KNOWHOW extracts gIoCs from CTI reports and encodes adversary behaviors as subject-verb-object triples. For instance, for T1570, KNOWHOW retrieves gIoCs such as `<adversary, use, ssh service>` and `<adversary, transfer, file>`. These precisely match the provenance graph derived from system events, including a *sh* process spawning an *sshd* service and subsequently writing files like */tmp/sshd* or */var/tmp/.>_*. Similarly, for T1036 (Masquerading) and T1222 (File and Directory Permissions Modification), KNOWHOW leverages gIoCs like `<adversary, create, system file name>` and `<adversary, modify, file permission>`. These gIoCs align with observed system events, such as a *touch* process creating the *yum.log* file, and then a *sh* process executing *chattr -i* and *chmod +x* commands on it. As for T1027 (Obfuscated Files or Information), KNOWHOW leverage gIoCs of `<adversary, execute, obfuscated file>` to detect the attacker's obfuscated executables like like */usr/bin/yzhulutzjr* and */usr/bin/zqjxlypgjo*. This semantic alignment between provenance graph and CTI enables high-fidelity TTP attribution.

Notably, this campaign employed some mimic and stealthy behaviors such as *Masquerading (T1036)* and *Obfuscated Files or Information (T1027)* to evade detection, which are precisely the kinds of behaviors that challenge conventional defense systems. For example, the attacker stored malicious binaries in hidden paths such as */tmp/.>_* and */var/tmp/.>_*, naming them *yum.log*, *apt.log*, and *gcc.sh* to mimic legitimate system files. Command-line obfuscation was also employed. Attackers executed common commands through seemingly random executable names such as */usr/bin/yzhulutzjr* and */usr/bin/zqjxlypgjo* to evade rule-based systems that rely on static patterns, e.g., known file paths or command keywords. Similarly, learning-based approaches struggle because these obfuscated behaviors are rare and lack sufficient training samples. Consequently, both approaches suffer from false negatives. If undetected, these techniques allow attackers to maintain silent persistence for weeks, eventually leading to full system compromise. In contrast, SYSARMOR leverages provenance analysis and ML-based detection to effectively identify these evasive activities despite their stealth.

C. RQ2: Scalability

1) End-to-End Throughput: To evaluate the end-to-end throughput of SYSARMOR, we measure the number of events processed per second by processing a fixed workload throught one agent because it is hard to simulate multiple agents generating real-time events to reach the peak throughput. We replay the collected events from files on one endpoint and

TABLE III: The throughput of SysArmor with different Flink job configurations. "eps" stands for events per second.

Job Configuration	Throughput (eps)
Job 01 + Job 02	12547
Job 01 + Job 03	12208
Job 01 + Job 02 + Job 03	9003

forward them to the SysArmor server. By measuring the time required to process all events, we obtain the upper-bound throughput for a single agent. The collected events are first sent to Apache Kafka for buffering, and then consumed by Apache Flink jobs for real-time processing. The first Flink job (Job 01: Event Normalization) normalizes raw auditd events into Sysdig format, and the subsequent jobs (Job 02: Rule-based Detection and Job 03: ML-based Detection) perform threat detection. We evaluate the throughput of individual Flink jobs, including *Job 02: Rule-based Detection* and *Job 03: ML-based Detection*. Then we evaluate the combined throughput when both jobs run concurrently.

Table III shows the throughput with different Flink job configurations. The throughput of Job 01 + Job 02 and Job 01 + Job 03 is similar, around 12,000 eps. When both detection jobs run concurrently, the throughput slightly decreases to around 9,000 eps. This is because that both jobs independently consume the same event stream from Kafka. Since they belong to different consumer groups, Kafka must deliver each event twice, effectively doubling broker egress traffic and disk I/O. This additional pressure on the Kafka cluster and network stack increases end-to-end processing overhead, which manifests as a drop in the overall throughput when both detection jobs are enabled simultaneously.

In our experiment, the average event generation rate for per endpoint is around 57 eps. Therefore, SysArmor can easily handle more than 150 endpoints in real-time with a single server. We further analyze the bottleneck of SysArmor and find a surprising result that Job 01 is the bottleneck of the entire pipeline because we deploy naive parsing logic in Job 01 without any optimization. We believe that with further optimizations on Job 01, the throughput of SysArmor can be significantly improved.

D. RQ3: Resource Consumption

To evaluate the resource consumption of SysArmor, we monitor the memory usage of SysArmor server during the experiment in Section IV-C. The integration of provenance analysis techniques may introduce additional memory overhead due to the need to maintain in-memory data structures for constructing and analyzing provenance graphs. Therefore, we specifically focus on measuring the memory consumption of the threat detection engine of SysArmor. In threat detection engine, Job 02 and Job 03 run in individual Flink containers, and each container is allocated with 8 GB memory. We timely monitor the memory usage of each container during the experiment. Figure 5 shows the result. We can observe that the

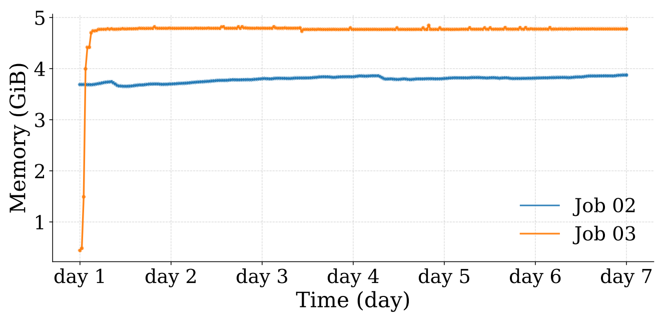

Fig. 5: Memory consumption of SysArmor server.

memory consumption of Job 02 is relatively low because the rule-based detection engine only needs to maintain a small set of rules and match them against incoming events. The Job 03 has an increased memory consumption during the first day due to the loading of NodLink model and KnowHow knowledge base and the cached suspicious provenance graph data structures. The memory consumption of both jobs remains stable after an initial warm-up period, with Job 02 consuming around 3.8 GB and Job 03 consuming around 4.8 GB. This indicates that SysArmor only requires moderate memory resources to handle real-time threat detection.

V. Takeaways

During our depolyment of SysArmor in the Third Research Institute of Ministry of Public Security, we found the key challenges and also collected the valuable feedbacks from the security analysts at the Third Research Institute of Ministry of Public Security. We summarized them as three important lessons as follows:

Lessons 1: Challenges in integrating provenance analysis into EDR systems. The primary challenge in deploying provenance-based EDR systems is efficiently processing large-scale provenance data in real-time. This involves handling complex workflows such as event collection, normalization, and hybrid detection using rule-based and ML engines. Surprisingly, we found that the current bottleneck in SysArmor is not the ML inference engine but the event normalization module, which transforms raw audit logs into standardized provenance events. This process is computationally intensive due to the complex parsing and transformation required for diverse data sources and formats. While optimizations like parallel and batch processing can mitigate this issue, it is crucial to carefully identify and address bottlenecks to ensure an efficient processing pipeline. We also find that different Flink jobs configurations also have impact on the overall system performance. It is crucial when we want to extend SysArmor with more detection modules in the future. These findings highlight the need for further research into optimizing provenance data processing pipelines to enable scalable and real-time threat detection in EDR systems.

Lessons 2: Limitations of existing ML-based provenance analysis. During the deployment of SysArmor, the security analysts from the Third Research Institute of Ministry of

Public Security find that the detection accuracy of ML inference is highly affected by the diversity of normal behaviors running on the endpoints. When the main jobs on the endpoints changes, it always introduce more false positives. However, to our best knowledge, existing provenance analysis techniques are not supported adaptive learning to continuously update the detection models according to the changing normal behaviors. Therefore, how to improve the robustness and stability of system against diverse production environments but do not loss the detection capability is crucial for large scale deployment in production environment.

Lessons 3: Gap between alert and incident response. Although SYSARMOR is able to interpret the alerts with TTPs and visualize the alerts in concise graph, there is still a gap between alert generation and incident response. Existing research on EDR systems mainly focuses on improving detection capability, while neglecting the operational needs of security teams. In practice, security analysts need more actionable insights and guided responses to effectively mitigate the detected threats. Therefore, future research should explore better automated response mechanisms that can translate alerts into concrete remediation actions. This would help close the loop from detection to response and enhance the overall effectiveness of EDR systems in real-world scenarios.

VI. DISCUSSION

SYSARMOR faces several limitations that warrant careful consideration for both current deployments and future development directions. These limitations represent opportunities for improvement and highlight areas where the system's capabilities could be enhanced through continued research and development efforts. We also propose potential solutions and future work to improve SYSARMOR.

Advanced Data Collection Interagations. Future work should explore integrating advanced data collection techniques [45]–[49] into SYSARMOR to enhance the efficiency and integrity of provenance data collection, thereby improving overall system performance and security monitoring capabilities. Recent work has focused on improving both the efficiency and effectiveness of endpoint data collection. Existing approaches fall broadly into two categories: software-based and hardware-based techniques. Software-based systems use mechanisms such as threadlets [45] and eBPF instrumentation [46] to intercept system calls efficiently, providing tamper-resistant logging with low performance overhead. Hardware-based designs instead leverage protected execution environments to offload portions of the logging pipeline, thereby enhancing both performance and integrity [47], [48].

Advanced Detection Algorithm Interagations. Future work should intergrate and explore more provenance-based detection algorithms that can enhance the system's threat detection capabilities. Recent research has proposed various provenance analysis techniques for APT detection and investigation [12], [50], [51]. SYSARMOR can intergrate these advanced detection algorithms as additional Flink jobs to enhance its detection capabilities. As the development of LLM, future work can also explore the integration of LLM-based techniques for provenance analysis and threat detection.

Scalability and Performance. Addressing scalability challenges for large-scale deployments is essential for supporting enterprise-scale security monitoring requirements. Distributed processing enhancement should focus on developing enhanced distributed processing capabilities for massive-scale environments with thousands of endpoints and millions of events per second. Future work can interagte data reduction techniques [25], [52], [53] and local rule-engine [54] to reduce the data volume needed to be processed in the central SYSARMOR system while maintaining detection effectiveness. And efficient event normalization techniques should be developed to parse and standardize diverse provenance data formats from different platforms and data sources, ensuring consistent processing and analysis across heterogeneous environments.

Operational Improvements. Future work should address operational challenges and improve usability to make SYSARMOR more accessible and manageable for organizations. Automated deployment development should focus on creating simplified deployment and configuration tools that reduce operational complexity and enable rapid setup in diverse environments. Security enhancements should also be prioritized to strengthen the system's resilience against attacks targeting the EDR infrastructure itself [55]. To make the investigation and response process more efficient, we can leverage LLM and develop agent to assist security analysts in alert triage and incident investigation [56].

Domain Expansion. Expanding SYSARMOR's applicability to other security domains would increase its value for organizations with diverse security monitoring needs. Recent work have proposed provenance-based techniques for Internet of Things (IoT) security [57], AI System security [58]–[60], and network security [61]. Future work can explore adapting SYSARMOR to these domains by integrating with relevant data and developing specialized algorithms tailored to the unique characteristics and threat landscapes of these environments.

A. Conclusion

Provenance-based EDR systems still face significant challenges in real-world deployment, including efficient data collection, real-time processing, and alert interpretation. SYSARMOR represents an attempt to address these challenges through a comprehensive framework that integrates advanced provenance analysis techniques into existing EDR systems. SYSARMOR introduces efficient data collection, real-time streaming processing, asynchronous hybrid detection with rule and ML engines, and alert interpretation and visualization. Through the deployment of SYSARMOR in a real-world enterprise environment for one week, we show the potential of provenance-based EDR systems to enhance endpoint security auditing. We also faces several limitations that warrant future research and development efforts to improve scalability, performance, and operational usability. We hope SYSARMOR serves as a foundation for further exploration and innovation in provenance-based endpoint security auditing.

References

[1] "Optimize endpoint security with edr," https://www.paloaltonetworks.com/cortex/endpoint-detection-and-response.

[2] "Endpoint detection and response (edr)," https://www.crowdstrike.com/en-us/resources/white-papers/endpoint-detection-and-response/.

[3] "Detect and respond at runtime," https://www.wiz.io/platform/wiz-defend.

[4] R. Uetz, M. Herzog, L. Hackländer, S. Schwarz, and M. Henze, "You cannot escape me: Detecting evasions of SIEM rules in enterprise networks," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 5179–5196. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/uetz

[5] X. Shen, Z. Li, G. Burleigh, L. Wang, and Y. Chen, "Decoding the mitre engenuity att&ck enterprise evaluation: An analysis of edr performance in real-world environments," 2024. [Online]. Available: https://arxiv.org/abs/2401.15878

[6] F. Dong, S. Li, P. Jiang, D. Li, H. Wang, L. Huang, X. Xiao, J. Chen, X. Luo, Y. Guo, and X. Chen, "Are we there yet? An Industrial Viewpoint on Provenance-based Endpoint Detection and Response Tools," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2396–2410. [Online]. Available: https://doi.org/10.1145/3576915.3616580

[7] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: Runtime provenance-based detector for advanced persistent threats," in *NDSS*, 2020.

[8] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 489–506.

[9] S. Li, F. Dong, X. Xiao, H. Wang, F. Shao, J. Chen, Y. Guo, X. Chen, and D. Li, "NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation." in *NDSS*, 2024.

[10] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance," 2023.

[11] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, "MAGIC: Detecting advanced persistent threats via masked graph representation learning," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 5197–5214. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/jia-zian

[12] T. Bilot, B. Jiang, Z. Li, N. El Madhoun, K. Al Agha, A. Zouaoui, and T. Pasquier, "Sometimes Simpler is Better: A Comprehensive Analysis of State-of-the-Art Provenance-Based Intrusion Detection Systems," in *Security Symposium (USENIX Sec'25)*. USENIX, 2025.

[13] Y. Meng, S. Li, J. Gui, P. Jiang, and D. Li, "Knowhow: Automatically applying high-level cti knowledge for interpretable and accurate provenance analysis," 2025. [Online]. Available: https://arxiv.org/abs/2509.05698

[14] H. Kaur, D. S. SL, T. Paul, R. K. Thakur, K. V. K. Reddy, J. Mahato, and K. Naveen, "Evolution of endpoint detection and response (edr) in cyber security: A comprehensive review," in *E3S Web of Conferences*, vol. 556. EDP Sciences, 2024, p. 01006.

[15] D. A. S. GEORGE, A. H. George, T. Baskar, and D. Pandey, "Xdr: the evolution of endpoint security solutions-superior extensibility and analytics to satisfy the organizational needs of the future," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 8, no. 1, pp. 493–501, 2021.

[16] "The linux audit framework." Redhat., accessed on October 1, 2023. [Online]. Available: https://github.com/linux-audit/

[17] "ETW events in the common language runtime," Microsoft, 2017, accessed on October 1, 2023. [Online]. Available: https://msdn.microsoft.com/en-us/library/ff357719(v=vs.110).aspx/

[18] "Sysdig." Sysdig., 2013, ccessed on October 1, 2023. [Online]. Available: https://sysdig.com

[19] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker *et al.*, "Prov-dm: The prov data model," *W3C Recommendation*, vol. 14, pp. 15–16, 2013.

[20] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *Network and Distributed Systems Security Symposium*, 2019.

[21] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: Runtime provenance-based detector for advanced persistent threats," *arXiv*, no. February, 2020.

[22] H. Ding, J. Zhai, Y. Nan, and S. Ma, "AIRTAG: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 373–390. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/ding-hailun-airtag

[23] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan, " SoK: History is a Vast Early Warning System: Auditing the Provenance of System Intrusions ," in *2023 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2023, pp. 2620–2638. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.10179405

[24] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "DEPCOMM: Graph Summarization on System Audit Logs for Attack Investigation," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 540–557.

[25] Y. Tang, D. Li, Z. Li, M. Zhang, K. Jee, X. Xiao, Z. Wu, J. Rhee, F. Xu, and Q. Li, "Nodemerge: Template based efficient data reduction for big-data causality analysis," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1324–1337. [Online]. Available: https://doi.org/10.1145/3243734.3243763

[26] J. Zeng, Z. L. Chua, Y. Chen, K. Ji, Z. Liang, and J. Mao, "Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics," in *Proceedings of the 28th Annual Network and Distributed System Security Symposium, NDSS*, 2021.

[27] P. Gao, F. Shao, X. Liu, X. Xiao, Z. Qin, F. Xu, P. Mittal, S. R. Kulkarni, and D. Song, "Enabling Efficient Cyber Threat Hunting With Cyber Threat Intelligence," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 193–204.

[28] K. Satvat, R. Gjomemo, and V. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021, pp. 598–615.

[29] Z. Li, J. Zeng, Y. Chen, and Z. Liang, "AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports," in *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 589–609. [Online]. Available: https://doi.org/10.1007/978-3-031-17140-6_29

[30] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 103–115. [Online]. Available: https://doi.org/10.1145/3134600.3134646

[31] Redhat, "The linux audit framework," 2017, https://github.com/linux-audit/.

[32] R. Gerhards *et al.*, "rsyslog," 2025. [Online]. Available: https://www.rsyslog.com/doc/

[33] D. Inc., "Vector: A lightweight, ultra-fast tool for building observability pipelines," 2025. [Online]. Available: https://vector.dev/

[34] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, no. 2011. Athens, Greece, 2011, pp. 1–7.

[35] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *The Bulletin of the Technical Committee on Data Engineering*, vol. 38, no. 4, 2015.

[36] T. F. Authors, "Detect security threats in real time," 2025. [Online]. Available: https://falco.org/

[37] T. L. Foundation, "Opensearch: Find the meaning in your data," 2025, https://opensearch.org/.

[38] T. P. G. D. Group, "Postgresql: The world's most advanced open source relational database," 2025, https://www.postgresql.org/.

[39] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces.* " O'Reilly Media, Inc.", 2011.

[40] React, "React," 2025. [Online]. Available: https://react.dev/

[41] J. Naor, D. Panigrahi, and M. Singh, "Online node-weighted steiner tree and related problems," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011, pp. 210–219.

[42] T. F. Authors, "Fastapi framework, high performance, easy to learn, fast to code, ready for production," 2025. [Online]. Available: https://fastapi.tiangolo.com/

[43] "The open source security platform," https://wazuh.com/.

[44] "MITRE ATT&CK," The MITRE Corporation, 2023, accessed on October 10, 2023. [Online]. Available: https://attack.mitre.org/

[45] P. Jiang, R. Huang, D. Li, Y. Guo, X. Chen, J. Luan, Y. Ren, and X. Hu, "Auditing frameworks need resource isolation: A systematic study on the super producer threat to system auditing and its mitigation," in *32nd USENIX Security Symposium (USENIX Security'23)*, 2023.

[46] R. Sekar, H. Kimm, and R. Aich, "eaudit: A fast, scalable and deployable audit data collection system," in *2024 IEEE Symposium on Security and Privacy (SP'24)*, 2024.

[47] A. Ahmad, S. Lee, and M. Peinado, "Hardlog: Practical tamper-proof system auditing using a novel audit device," in *2022 IEEE Symposium on Security and Privacy (SP'22)*, 2022.

[48] C. Zhang, J. Zeng, Y. Zhang, A. Ahmad, F. Zhang, H. Jin, and Z. Liang, "The hitchhiker's guide to high-assurance system observability protection with efficient permission switches," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (CCS'24)*, 2024.

[49] P. Jiang, H. Jiang, R. Huang, H. Lei, Z. Zhong, S. Zhang, Y. Ren, N. Jia, X. Hu, Y. Guo *et al.*, "Dpuaudit: Dpu-assisted pull-based architecture for near-zero cost system auditing," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA'25)*, 2025.

[50] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier, "Orthrus: achieving high quality of attribution in provenance-based intrusion detection systems," in *Proceedings of the 34th USENIX Conference on Security Symposium*, ser. SEC '25. USA: USENIX Association, 2025.

[51] D. Sun, J. Zhang, J. Xu, Y. Zheng, Y. Tian, and Z. Li, "From alerts to intelligence: A novel llm-aided framework for host-based intrusion detection," 2025. [Online]. Available: https://arxiv.org/abs/2507.10873

[52] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, and G. Jiang, "High fidelity data reduction for big data security dependency analyses," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 504–516. [Online]. Available: https://doi.org/10.1145/2976749.2978378

[53] H. Ding, S. Yan, J. Zhai, and S. Ma, "ELISE: A storage efficient logging system powered by redundancy reduction and representation learning," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3023–3040. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/ding

[54] F. Dong, L. Wang, X. Nie, F. Shao, H. Wang, D. Li, X. Luo, and X. Xiao, "DISTDET: A Cost-Effective distributed cyber threat detection system," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 6575–6592. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/dong-feng

[55] K. Alachkar, D. Gaastra, E. Barbaro, M. van Eeten, and Y. Zhauniarovich, "Eviledr: repurposing edr as an offensive tool," in *Proceedings of the 34th USENIX Conference on Security Symposium*, ser. SEC '25. USA: USENIX Association, 2025.

[56] K. Mukherjee and M. Kantarcioglu, "Llm-driven provenance forensics for threat investigation and detection," 2025. [Online]. Available: https://arxiv.org/abs/2508.21323

[57] Q. Wang, W. U. Hassan, A. Bates, and C. A. Gunter, "Fear and logging in the internet of things," in *Network and Distributed System Security Symposium*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3673561

[58] "Provenance-Based interpretation of Multi-Agent information analysis," in *12th International Workshop on Theory and Practice of Provenance (TaPP 2020)*. USENIX Association, Jun. 2020. [Online]. Available: https://www.usenix.org/conference/tapp2020/presentation/friedman

[59] R. Souza, A. Gueroudji, S. DeWitt, D. Rosendo, T. Ghosal, R. Ross, P. Balaprakash, and R. F. da Silva, "Prov-agent: Unified provenance for tracking ai agent interactions in agentic workflows," 2025. [Online]. Available: https://arxiv.org/abs/2508.02866

[60] S. Li, Z. Zhang, H. Jia, Y. Guo, X. Chen, and D. Li, " Query Provenance Analysis: Efficient and Robust Defense Against Query-Based Black-Box Attacks ," in *2025 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 1641–1656. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00072

[61] Z. Liu, J. Mao, J. Zeng, J. Li, Q. Lin, J. Liu, J. Zhuge, and Z. Liang, "Provguard: Detecting SDN control policy manipulation via contextual semantics of provenance graphs," in *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society, 2025. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/provguard-detecting-sdn-control-policy-manipulation-via-contextual-semantics-of-provenance-graphs/