MuSAR: Multi-Step Attack Reconstruction from Lightweight Security Logs via Event-Level Semantic Association in Multi-Host Environments

Yang Liu*
Xi'an Jiaotong University
Xi'an, China
yangliu@xjtu.edu.cn

Zisen Xu*
Xi'an Jiaotong University
Xi'an, China
zisenxu2000@163.com

Zian Luo†
Xi'an Jiaotong University
Xi'an, China
1342011793@stu.xjtu.edu.cn

Jin'ao Shang
Xi'an Jiaotong University
Xi'an, China
jinao_s@stu.xjtu.edu.cn

Shilong Zhang
Xi'an Jiaotong University
Xi'an, China
zsl846296479@stu.xjtu.edu.cn

Haichuan Zhang
University of Science and Technology of China
Hefei, China
zhanghaichuan@mail.ustc.edu.cn

Ting Liu†
Xi'an Jiaotong University
Xi'an, China
tingliu@mail.xjtu.edu.cn

***Abstract*—Multi-step attacks challenge security analysts in reconstructing attack sequences from extensive multi-host log data. Existing attack reconstruction approaches rely heavily on computationally intensive audit logs and provenance analysis, limiting their practicality in multi-host environments. We present MuSAR, a framework for real-time reconstruction of multi-step attacks in multi-host environments using lightweight security logs (network alarms and application logs). First, security logs are consolidated into inter-host and intra-host security events through semantic analysis, respectively. Then, these security events are mapped to unified attack lifecycle stages through MITRE ATT&CK framework integration. Finally, a heuristic algorithm is implemented to identify potential multi-step attacks and reconstruct complete attack sequences based on event-level semantic associations. Evaluation on the CPTC2018 dataset and a multi-step attack simulation dataset demonstrates MuSAR's effectiveness in identifying attack-related traces and reconstructing multi-step attacks, achieving an average recall of 93.48% and F1-score of 94.39%, respectively, outperforming state-of-the-art methods in attack investigation and reconstruction in multi-host environments.**

***Index Terms*—Lightweight Security Logs, Semantic Alignment, Event-level Semantic Association, Attack Reconstruction.**

I. Introduction

The network security landscape has become increasingly critical, as escalating threats continue to undermine the effectiveness of traditional defensive measures. Among these, multi-step or multi-stage attacks [1], which comprise a series of logically interrelated steps, pose significant obstacles for analysts, particularly in threat detection and investigation [2]. Specifically, the attack process is decomposed into multiple discrete attack steps to accomplish the ultimate attack goal, where each step is designed to achieve a specific objective, such as SQL injection or credential cracking [3]. These attack steps can be conducted slowly on numerous hosts, making it difficult to recover the whole attack process from fragmented traces dispersed across multiple network zones over a long period of time [4].

Attack reconstruction serves as a fundamental approach for tracing multi-step attacks, enabling analysts to understand the sequence of interrelated malicious activities systematically. Many existing studies focus on a limited variety of log types [5], [6], which results in insufficient analysis of fragmented traces, thereby hindering satisfactory reconstruction. Furthermore, recent works predominantly leverage audit logs to conduct provenance analysis in single-host scenarios [7]–[9]. These approaches focus on the detection of anomalous entities (e.g., processes and files) rather than the modeling of attack actions across multiple hosts (e.g., lateral movement), making it challenging to identify critical attack steps precisely, which reflect the tactics and techniques employed by attackers. Additionally, despite various proposed solutions, such as distributed detection [10] and optimized storage [11], [12], the substantial computational complexity and storage requirements associated with audit logs continue to hinder their widespread deployment in multi-host production environments [13].

To defend against multi-step attacks, network administrators deploy diverse security devices at critical points. These devices generate heterogeneous logs with distinct formats and semantics, each capturing specific aspects of the attack lifecycle. For instance, network-based Intrusion Detection Systems (IDS) generate alarms for network-level attacks but cannot detect host-based privilege escalation attempts — activities that application logs readily capture. Unfortunately, to the best of our knowledge, application logs, particularly command history logs (e.g., `.bash_history`), are rarely incorporated into attack reconstruction due to their semi-structured nature and semantic dependencies on context. While the integration of diverse security logs enables comprehensive attack visibility, the semantic heterogeneity and distributed nature of these logs pose significant challenges for effective association analysis

* Equal contribution. † Corresponding Authors.

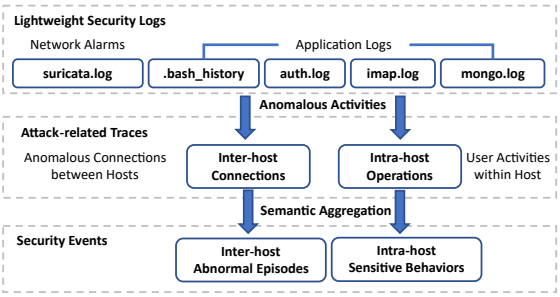

Fig. 1: Security event generation process in multi-host environments. MuSAR first extracts two types of attack-related traces from lightweight security logs. These traces are then aggregated into inter-host and intra-host security events, respectively. We have considered five categories of representative logs in this paper.

and accurate attack reconstruction.

In this paper, we introduce MuSAR, a framework for **Mu**lti-**S**tep **A**ttacks **R**econstruction in multi-host environments through heuristic analysis of event-level semantic associations derived from lightweight security logs. Fig. 1 depicts the systematic process of security event generation from lightweight security logs. MuSAR automates the extraction of security events and identification of multi-step attacks without requiring any attack-related ground truth. Furthermore, the lightweight security logs enable MuSAR to identify and reconstruct multi-step attack processes with minimal overhead, assisting analysts in performing real-time threat analysis. MuSAR consists of three steps: (1) Lightweight security logs are consolidated into two types of log-level traces (inter-host connections and intra-host operations) with structured representation, which are then aggregated into events that reflect the underlying attack actions; (2) Heterogeneous events are semantically aligned through mapping to standardized attack stages within a unified attack lifecycle; (3) A heuristic algorithm is developed to reconstruct and visualize potential multi-step attacks through event-level semantic association.

We evaluate MuSAR's performance in multi-step attack reconstruction using the CPTC2018 dataset [14]. Additionally, by referring to the tactics and techniques in the CPTC2018 dataset, we replicate two multi-step attack scenarios in a controlled testbed and construct a multi-step attack simulation dataset (including both lightweight security logs and audit logs) to facilitate comparative analysis with State-Of-The-Art (SOTA) methods, including SAGE [5], ATLAS [15], and MAGIC [16]. Evaluation results demonstrate that MuSAR effectively identifies attack-related traces and reconstructs multi-step attacks, achieving an average of 93.48% recall and 94.39% F1-score, outperforming SOTA methods in attack investigation and reconstruction. Furthermore, the attack graphs visualized by MuSAR accurately restore the multi-step attack processes, providing valuable insights for analysts to comprehend the attackers' intent.[1] Our main contributions are summarized as follows:

- We present MuSAR, a framework that heuristically automates the extraction of security events and the reconstruction of potential multi-step attacks across multi-host environments from lightweight security logs, without requiring any attack-related ground truth. We have also developed a real-time interactive analysis system integrating MuSAR's core functionalities, demonstrating MuSAR's effectiveness.
- We design two rule sets (588 rules in total) to aggregate lightweight security logs into inter-host and intra-host security events, respectively. These events are then mapped to the MITRE ATT&CK framework's stages through semantic analysis and 35 predefined stage mapping rules, creating a unified attack lifecycle through event-level semantic association.
- We construct a multi-step attack simulation dataset incorporating network alarms, application logs, and audit logs with detailed ground truth annotations for accurate performance evaluation and comparative analysis with SOTA methods. Evaluation results demonstrate MuSAR's capability to reconstruct key attack steps with high accuracy while maintaining minimal computational overhead.

II. Motivating Example

Fig. 2(a) illustrates a real-world multi-step attack scenario where an attacker orchestrates a sophisticated campaign across multiple network zones. Specifically, after compromising host-1, the attacker systematically employs reconnaissance, vulnerability exploitation, and lateral movement tactics to compromise remaining hosts (host-2 and host-3) en route to host-4. Subsequently, the attacker exploits an anonymous login vulnerability on File Transfer Protocol (FTP) to exfiltrate sensitive data from host-4, demonstrating an attack path that traverses both the internet and internal network zones. We collected network traffic and application logs from the involved hosts and generated network alarms using Suricata [17].

As shown in Fig. 2(b), network alarms captured port scanning, vulnerability discovery, and FTP anonymous login activities during the progression from host-3 to host-4, yet failed to identify critical actions such as data exfiltration. While these alarms indicated potential root privilege acquisition during the vulnerability discovery stage, it is insufficient to confirm such privileged activities through network alarms alone. In contrast, application logs (i.e., `.bash_history`) from host-3 provided crucial complementary evidence. As shown in Fig. 2(b), the left dashed block documents the attacker's privilege escalation sequence: downloading the malicious script `dirty.c` to a compromised host, then compiling and executing it to exploit `CVE-2022-0847`. These application logs filled critical gaps in understanding the vulnerability scan stage. The right dashed block reveals file transfer activities through FTP anonymous login, which are undetectable via network-based monitoring.

[1]MuSAR and the datasets are available at https://anonymous.4open.science/r/MuSAR.

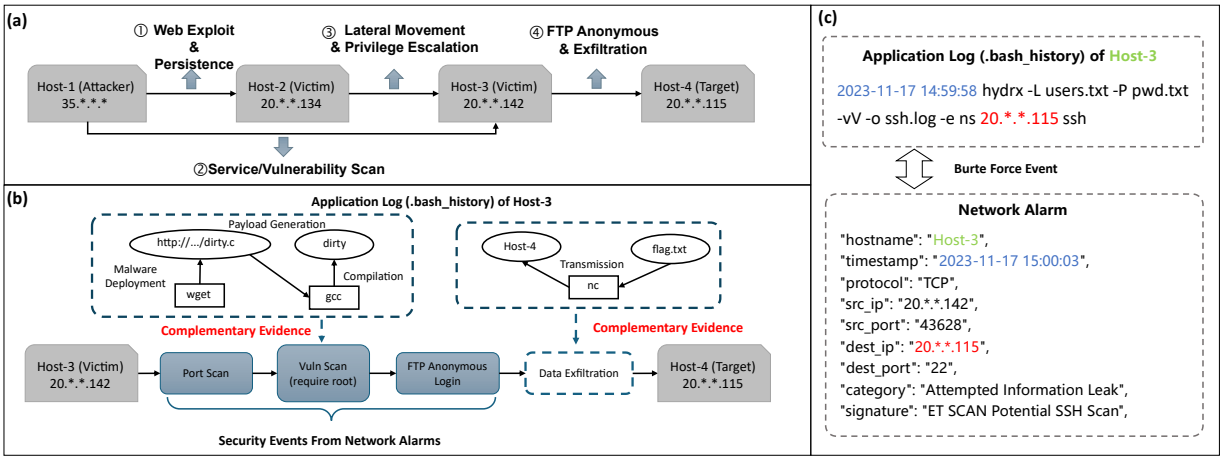

Fig. 2: Anatomy of a real-world multi-step attack. (a) Complete attack procession across four compromised hosts. (b) Security events between host-3 and host-4, where dashed blocks indicate host-based evidence from application logs of host-3, complementing network-based detections (solid blocks). (c) Semantic correspondence between network alarms and application logs (e.g., `.bash_history`) documenting a brute force event, with matching fields highlighted in distinct colors. Note that the command `hydrx` is a deliberate obfuscation of `hydra`, employed by adversaries to evade signature-based detection. The strategy to handle unseen commands is discussed in Appendix E3.

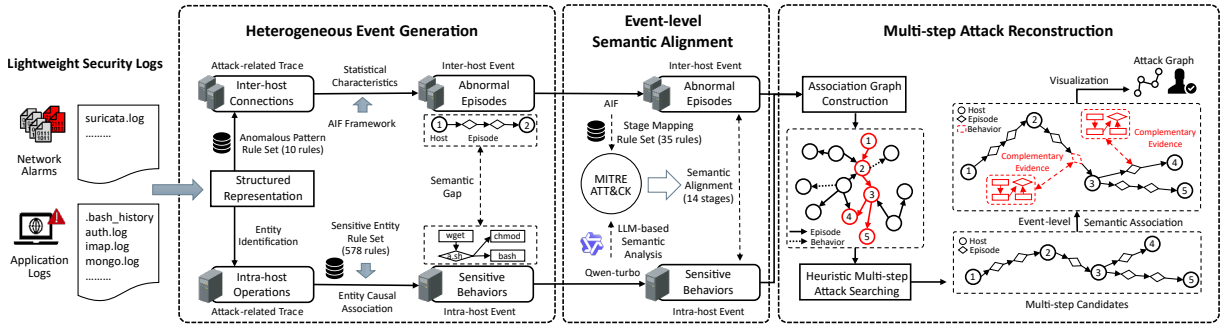

Fig. 3: Overview of the MuSAR framework. (1) Lightweight security logs (network alarms and application logs) are consolidated into two types of log-level traces (inter-host connections and intra-host operations), which are further aggregated into security events (inter-host abnormal episodes and intra-host sensitive behaviors, Sec. III-A). (2) Mapping rules from events to attack stages are established to achieve event-level semantic alignment (Sec. III-B). (3) A heuristic algorithm is implemented to reconstruct potential multi-step attacks through event-level semantic association (Sec. III-C).

From the example, we gain several key insights: (1) Different types of logs have distinct observable scopes and inherent limitations, making comprehensive attack detection challenging (e.g., network-based monitoring cannot capture intra-host privilege escalation activities). (2) Attack actions may appear across different log types while preserving semantic consistency. As depicted in Fig. 2(c), both network alarms and application logs can capture a brute force event with similar semantic elements. [2] (3) Attackers typically execute lateral movement through multiple compromised hosts to reach their ultimate targets, exhibiting a hop-based attack pattern. Therefore, accurate reconstruction of multi-step attack sequences necessitates the systematic integration of fragmented attack-related traces across diverse log sources.

[2] `hydrx` serves as an example of a variant of a known attack tool `hydra`.

III. Methodology

MuSAR Framework. MuSAR is a framework for attack reconstruction in multi-host environments that identifies multi-step attacks based on event-level semantic association across lightweight security logs. As presented in Fig. 3, MuSAR consists of three main steps. First, MuSAR processes and transforms lightweight security logs into structured representations and aggregates them into security events that capture the underlying attack actions. Second, based on the MITRE ATT&CK framework [18], MuSAR establishes a unified attack lifecycle to achieve event-level semantic alignment. Third, MuSAR employs a heuristic algorithm to reconstruct and visu-

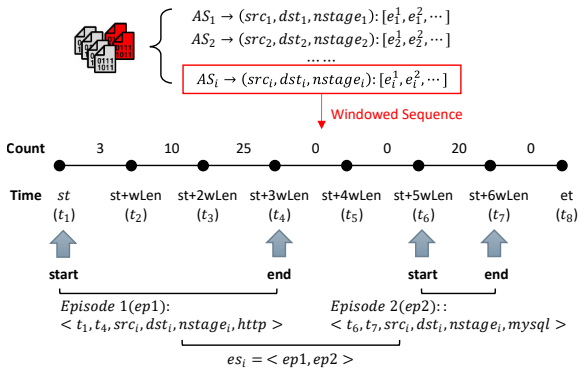

Fig. 4: Aggregation of two inter-host abnormal episodes from the windowed **A**bnormal **S**equences (AS) between src_i and dst_i within stage $nstage_i$, where the start time is st and the end time is et. Episode 1 starts at t_1 with an alarm count of 3 and spans three windows ($3wLen$) until t_4, with `http` as the most frequently targeted service. Episode 2 occurs within a single time window, targeting the `mysql` service.

alize potential multi-step attacks through event-level semantic association.

Threat Model and Assumptions. MuSAR focuses on multi-host environments where attackers breach the network boundaries to infiltrate the internal network and compromise the target by leveraging previously compromised hosts. Similar to provenance-based approaches [7], [15], we assume that all involved hosts are vulnerable only to software flaws, excluding hardware trojans and side-channel attacks. Although some hosts in access zones or demilitarized zones use dynamic IP addresses, critical hosts typically maintain static IP addresses to ensure service stability (e.g., authentication, FTP, web, and proxy services). Therefore, we choose IP addresses as reliable identifiers for hosts involved in attacks. Additionally, Our analysis relies on the integrity of lightweight security logs. While sophisticated attackers may attempt to erase their traces, existing mechanisms can protect log integrity [19]–[21].

A. Heterogeneous Event Generation

This section introduces the structured representation of two types of log-event traces (inter-host connections and intra-host operations) and details the generation process of the corresponding security events (inter-host abnormal episodes and intra-host sensitive behavior), respectively.

1) Inter-host Abnormal Episode: Inter-host abnormal episodes represent consolidated sequences of inter-host connections that characterize related security events between communicating hosts [22]. Network alarms, the predominant logging mechanism for inter-host connections [6], [23], encompass critical attributes including IP addresses, port numbers, timestamps, event signatures, and protocol-specific metadata. Based on SAGE [5], which effectively aggregates network alarms into episode sequences to capture the attack actions, MuSAR first represents an inter-host connection as $e =< t, src, dst, serv, nstage, sev >$. Here, t denotes the timestamp. src and dst represent the IP addresses of the source and destination, respectively. $serv$ is the corresponding service name mapped from the destination port number through the IANA table [24]. $nstage$ represents the attack stage mapped from the alarm signature through the Action-Intent-Framework (AIF) [25], which categorizes the attack lifecycle into 35 distinct stages according to the detection rules of open-source IDS, and sev indicates the severity.

Then, MuSAR extends SAGE by incorporating three additional host-based application logging sources — authentication logs (`auth.log`), email logs (`imap.log`), and database logs (`mongo.log`) — to identify potential anomalous connections undetectable through network alarms alone. However, application logs capture both benign and malicious activities and often lack standardized descriptive fields, complicating automated analysis without semantic context. Notably, these undetected activities often manifest as seemingly legitimate operations, such as lateral movement and shell establishment, enabling them to evade network-based IDS detection. To address this challenge, MuSAR leverages Wazuh [26], an open-source eXtended Detection and Response (XDR) platform, and enhances its detection rules to identify subtle anomalous patterns in application logs. Then, MuSAR augments these identified connections with semantic-derived attributes to establish a unified representation compatible with network alarm formats (see Appendix A1). The combination of network alarms and semantically enriched host-based connections forms a comprehensive inter-host connection dataset.

Finally, MuSAR gathers the inter-host connections into **A**bnormal **S**equences (AS) on the basis of (src, dst, $nstage$) triplet and partitions each AS into windowed sequences using a predefined window length parameter $wLen$. The i-th sequence consists of inter-host connections with timestamps within the interval $[st + (i-1) \times wLen,\ st + i \times wLen]$, where st denotes the starting timestamp of the AS. As illustrated in Fig. 4, the boundaries of abnormal episodes are determined by analyzing trends of alarm counts within each windowed sequence. Specifically, an increase from zero marks the start of an episode, while a drop to zero indicates its end. We represent an episode as $ep =< st, et, src, dst, nstage, mServ >$, where st and et are the start and end timestamps, $mServ$ is the most frequently targeted service name. Finally, the abnormal episodes generated by each (src, dst) pair are sorted by start timestamp to obtain the episode sequences, denoted as $es(src, dst) = \{ep_1, ep_2, \cdots, ep_n\}$.

2) Intra-host Sensitive Behavior: Intra-host sensitive behaviors, which characterize security events occurring within a host, consist of causally associated intra-host operations that are primarily derived from historical command logs (e.g., `.bash_history`). For instance, a sequence of operations such as downloading, modifying, and executing a malicious script constitutes an intra-host sensitive behavior. Inspired by the insight that causally associated intra-host operations typically share common entities (e.g., commands or files), we propose a novel method to extract semantic entities from his-

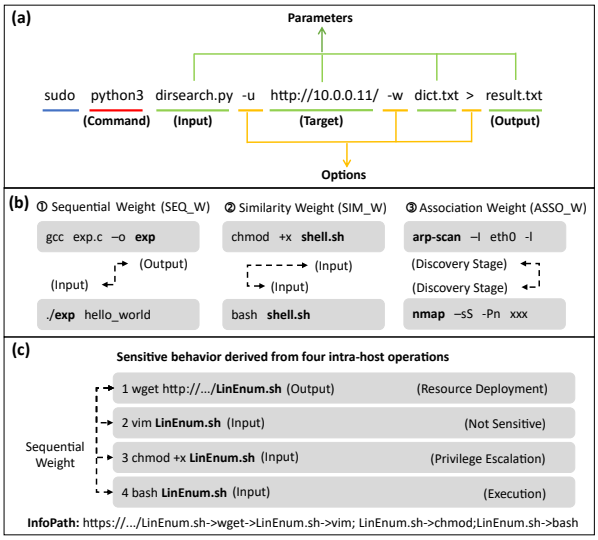

Fig. 5: Intra-host sensitive behavior aggregation and semantic alignment. (a) Syntax parsing and entity identification for an intra-host operation. (b) Three types of weights for quantifying operational associations, with sequential weights representing the strongest causal connections and association weights the weakest. (c) A sensitive behavior bh in the [Collection] stage, derived from four sequential intra-host operations with core semantics encapsulated as $InfoPath$.

torical command logs and identify sensitive behaviors through entity-based causal association analysis.

Structured Representation. MuSAR uses the open-source tool Bashlex [27] to parse the shell syntax structure, extract core components, and analyze contextual semantics to identify critical entities (command, input, output, and target), as illustrated in Fig. 5(a). Based on the mapping criteria between anomalous Linux commands and five specific attack stages proposed by Uptycs [28], we construct a comprehensive rule set (578 rules) for identifying sensitive entities (e.g., sensitive commands like `nmap` and sensitive files like `/etc/passwd`) and extend the criteria to encompass 14 attack stages of the MITRE ATT&CK framework [18] (see Appendix A2). Leveraging the extended criteria and the rule set, MuSAR classifies each intra-host operation as either sensitive or benign and determines its attack stage. Finally, the structured representation of an intra-host operation is denoted as $c =< t, host, command, input, output, target, isSens, stage >$, where t is the timestamp, $host$ is the host IP address, $command$ represents the executable command. $input$, $output$, and $target$ correspond to the relevant entities involved. $isSens$ indicates sensitivity and $hstage$ represents the attack stage.

Causal Association. The causal relationships between entities are investigated to aggregate related operations into sensitive behaviors. As illustrated in Fig. 5(b), we establish three categories of weights to quantify these relationships.

Algorithm 1: Aggregation of Sensitive Operations

Input: Time-sorted Operations $\mathcal{C} = \{c_1, c_2, \cdots, c_N\}$, Constants: $SEQ\_W, SIM\_W, ASSO\_W$
Output: Behavior Sequence bs

1 **Function** *BEHAVIOR_AGGREGATE($\mathcal{C}$)*
2 $bs \leftarrow [\,]$
3 $bh\_id \leftarrow [None] \times LENGTH(\mathcal{C})$
4 **for** *each* $c_i \in \mathcal{C}$ **do**
5 **if** $i == 0$ ***and*** $c_i.isSens$ **then**
6 $bh\_id[0] \leftarrow new\_id$
7 **else**
8 $max\_score \leftarrow 0$
9 **for** *each* $c_j \in \{c_{i-1}, \cdots, c_1\}$ **do**
10 $score \leftarrow CALC\_SCORE(c_i, c_j)$
11 **if** $score > max\_score$ **then**
12 $bh\_id[i] \leftarrow bh\_id[j]$
13 $max\_score \leftarrow score$
14 **end**
15 **end**
16 **end**
17 **end**
18 **for** *each* $id \in bh\_id$ **do**
19 $bh \leftarrow CLUSTERS\_BY\_SAMEID(bh\_id)$
20 $bs.append(bh)$
21 **end**
22 **return** bs
23 **Function** *CALC_SCORE(c_i, c_j)*
24 $weight \leftarrow CAUSAL\_ANALYSIS(c_i, c_j)$
25 $score \leftarrow weight/(|i-j|)$
26 **return** $score$

Sequential weight reflects the correlation between input and output entities, while similarity weight indicates operations sharing the same input entity. Association weight characterizes operations of identical command types without direct input-output dependency. Then, we aggregate related operations based on their inter-operation association scores using Algorithm 1. To clarify the algorithm's logic, we define its two helper functions:

- $CAUSAL\_ANALYSIS(c_i,\ c_j)$ examines the entities of operations c_i and c_j. It determines their relationship type and returns the single corresponding constant weight (SEQ_W, SIM_W, $ASSO\_W$). If no relationship exists, it returns zero.
- $CLUSTERS\_BY\_SAMEID(bh\_id)$ is a straightforward grouping function that collects all operations assigned the same behavior ID into a single cluster.

The algorithm proceeds as follows: (1) Each operation is initially assigned a behavior ID of "None", indicating it does not yet belong to any sensitive behavior (Line 3). (2) Operations are then processed sequentially. If the first operation is classified as sensitive, we assign it a new behavior ID; otherwise, we proceed to the next (Lines 5-6). (3) For each subsequent operation, association scores are calculated

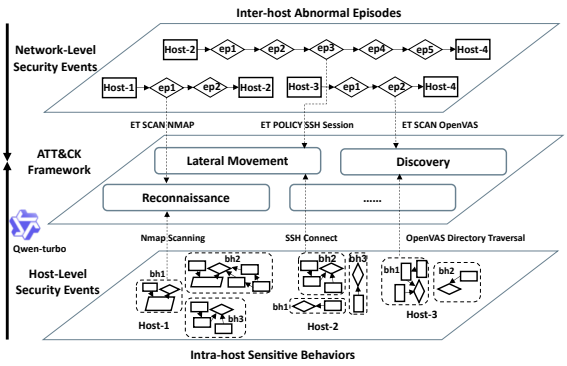

Fig. 6: Semantic alignment between inter-host episodes and intra-host behaviors under the MITRE ATT&CK framework.

to evaluate its relationship with previous operations in reverse chronological order and assign each operation the sensitive behavior that exhibits the highest score (Lines 7-16). (4) After traversal, intra-host operations sharing the same behavior ID are aggregated into sensitive behaviors (Lines 18-21). Moreover, a decay is incorporated in the calculation that diminishes as the distance increases, to prevent spurious associations between distant operations (Lines 23-26).

Each behavior, identified through Algorithm 1, is formally represented as $bh =< st, et, host, target, InfoPath >$, where st and et denote the behavior's start and end timestamps, respectively. $host$ specifies the host's IP address, while $target$ represents the IP addresses of the other hosts involved in the behavior. $InfoPath$ captures the behavior's semantic essence through a condensed representation of entity causal relationships, tracking information flow from input through command execution to output, as shown in Fig. 5(c). For a specific host h, all identified behaviors are chronologically ordered to construct its behavior sequence $bs(h) = \{bh_1, , bh_2, , \cdots , , bh_n\}$.

B. Event-level Semantic Alignment

As illustrated in Fig. 6, security events are characterized from complementary perspectives — inter-host abnormal episodes (Sec. III-A1) at the network level and intra-host sensitive behaviors (Sec. III-A2) at the host level — jointly capturing attack actions across different stages of the attack lifecycle. For example, both episodes containing `ET SCAN NMAP` signatures and behaviors using `nmap` are classified under the [Reconnaissance] stage. However, the inherent semantic gap between these perspectives poses challenges in establishing event-level semantic associations. To address this, we align these heterogeneous events by mapping them to corresponding attack stages within a unified attack lifecycle framework.

Our approach involves two key steps: (1) We develop mapping rules (see Appendix A3) to correlate 35 techniques from the AIF framework with 14 tactics from the ATT&CK framework for abnormal episodes. (2) To address the variability in sensitive host behaviors (e.g., shell command histories), we employ the Qwen-turbo Large Language Model (LLM) [29] to map behavior sequences to ATT&CK attack stages. The model processes structured prompts containing operational context and representative examples, outputting stage labels that extend each behavior bh to $bh =< bh, bstage >$. Specifically, we adopt a two-phase LLM-based semantic alignment for intra-host behaviors. By first filtering to the top three candidate tactics and then performing precise classification using in-context examples, we improve classification accuracy and robustness. Full prompt designs are detailed in Appendix A3.

To improve alignment accuracy and effectively mitigate the many-to-many relationships arising from the inherent complexity of attack stages during event-level semantic alignment, we implement a two-pronged approach: (1) We reuse the stage information of episodes in the AIF framework with finer granularity, allowing each intra-host sensitive behavior to associate with multiple potential inter-host abnormal episodes. (2) We determine optimal one-to-one mappings between episodes and behaviors by semantic context from raw logs. For example, during the analysis of a [Discovery] stage episode, multiple sensitive behaviors may be potential matches. If network alarms contain `ET SCAN Nikto` signatures, the episode is more likely associated with scanning activities using `nikto` rather than directory traversal operations using `dirsearch`.

C. Multi-step Attack Reconstruction

1) Overview of Multi-step Attack: Multi-step attacks comprise logically interconnected, sequential actions executed to achieve specific attack objectives [1]. We define a hop as a host pair encompassing one or more attack stages. Consequently, multi-step attacks can be represented as $MsA = \{hop_1, hop_2, \cdots , hop_n\}$, where $n \geq 1$. Based on the number of hosts involved, we classify multi-step attacks into multi-hop attacks ($n > 1$) and single-hop attacks ($n = 1$). As illustrated in Fig. 7(a), a multi-hop attack involves at least one compromised intermediary host, where the destination IP address in each hop becomes the source IP address in the subsequent hop. In contrast, a single-hop attack (Fig. 7(b)) consists of a source IP address directly targeting a destination IP address through multiple distinct attack stages without intermediary hosts. During attack reconstruction, some observed single-hop attacks may represent isolated fragments of a multi-hop attack campaign due to the missing of intermediate traces.

2) Heuristic Multi-step Attack Searching: The motivating example in Sec. II reveals two key characteristics of multi-step attacks: (1) They typically exhibit a hop-based pattern with compromised hosts serving as intermediaries. (2) They unfold chronologically with attack tactics and techniques evolving progressively. With these insights, MuSAR employs a heuristic algorithm that reconstructs potential multi-step attack sequences by leveraging event-level semantic associations among fragmented traces, as depicted in Fig. 8. The algorithm comprises four essential phases:

(a) Candidate Searching. MuSAR first constructs a directed association graph to represent spatial relationships

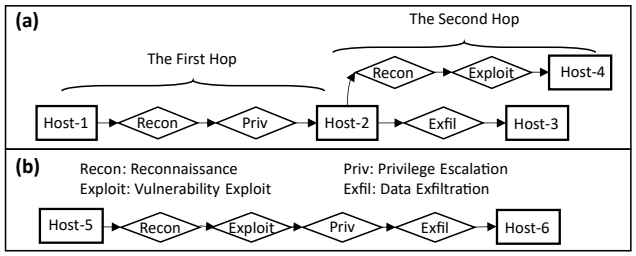

Fig. 7: Two types of multi-step attacks. (a) A two-hop attack involving four hosts. (b) A single-hop attack progression through four distinct attack stages.

among security events (i.e., inter-host abnormal episodes and intra-host sensitive behaviors). In this graph, nodes represent hosts involved in episodes and behaviors with non-empty target entities, connected by directed edges, which indicate the direction of the attack. As shown in Fig. 8(a), the algorithm initiates from nodes with zero in-degree and employs Depth-First Search (DFS) to identify multi-hop attack candidates containing more than two hops. The traversal terminates upon reaching either a node with zero out-degree or detecting a cycle pattern. Subsequently, we extract single-hop candidates from the remaining isolated associations that exhibit more than four distinct attack stages, where at least one stage indicates potential adversarial control over the target host.

(b) Feasibility Verification. MuSAR validates all candidates across three dimensions beyond spatial relationships to eliminate false positives, as shown in Fig. 8(b). Initially, MuSAR employs subset analysis to identify non-redundant longest multi-step attack sequences. Given two multi-step attack candidates MsA_1 and MsA_2, we remove MsA_1 if the attack sequence it contains is a subset of MsA_2. Next, temporal causality is verified by ensuring that for each MsA, the earliest timestamp of each hop precedes the latest timestamp of its subsequent hop, guaranteeing a logical progression of attack steps. Finally, for multi-hop attack candidates, each intermediate hop must contain evidence of successful target host access; hops exhibiting only reconnaissance activities are deemed invalid.

(c) Similarity Merging. Multi-hop attack candidates sharing identical prefix (first N hops) or suffix (last N hops) sequences likely belong to the same attack process. MuSAR merges these candidates to achieve a more comprehensive attack reconstruction. As illustrated in Fig. 8(c), the prefix merging process combines the subsequent hops of two multi-hop attack candidates MsA_1 and MsA_2 when the security events contained in their first N hops are identical, forming a new multi-hop attack sequence MsA. An analogous process is implemented for suffix merging. For single-hop attack candidates, MuSAR merges those with shared source or destination endpoints. These merged sequences constitute the final reconstructed multi-step attack set.

(d) Semantic Supplementation. MuSAR enriches the context of multi-step attacks by correlating standalone intra-host sensitive behaviors with inter-host abnormal episodes through attack stage matching. For each inter-host episode in the identified multi-step attacks, MuSAR identifies sensitive behaviors that share the same attack stage. Once matched, MuSAR employs the optimization techniques detailed in Sec. III-B to integrate the most relevant behavior into the corresponding episode as contextual information, as illustrated in the top-right dashed block of Fig. 8(d). Unmatched behaviors are processed according to two criteria: (1) Behaviors with non-empty target entities and attack stages not present in inter-host abnormal episodes are incorporated as new episodes within the multi-step attacks, as shown in the bottom-right dashed block of Fig. 8(d). (2) Behaviors without external interactions are incorporated as node attributes to enhance attack context for comprehensive security analysis.

3) Attack Graph Visualization: The reconstructed multi-step attacks by MuSAR are transformed into attack graphs using Graphviz [30]. Each graph provides a visual representation of identified security events and their corresponding attack tactics and techniques, enabling analysts to quickly trace attack progression patterns. Fig. 8(e) presents the simplified attack graph reconstructed from the case in Sec. II, demonstrating MuSAR's effectiveness in identifying critical attack steps and reconstructing missing evidence from application logs. Specifically, in the third hop from host-3 to host-4, MuSAR reconstructs the complementary evidence of `DATA DELIVERY` using the `nc` command, denoted as the last event with $InfoPath$ as `Host-4 -> nc`. Furthermore, MuSAR incorporates the behavior of privilege escalation on host-3 as node attributes since it represents a local operation without external interactions, highlighting potential security implications for further analysis.

IV. Evaluation

We evaluate MuSAR against SOTA approaches through the following research questions:

RQ1. How does the integration of application logs enhance multi-step attack reconstruction compared to network-alarm-only approaches? (Sec. IV-B)

RQ2. Can MuSAR provide interpretable attack strategies and accurately reconstruct multi-step attack sequences? (Sec. IV-C)

RQ3. Can MuSAR outperform SOTA approaches in multi-host environments? (Sec. IV-D)

RQ4. What is MuSAR's efficiency and scalability in production environments? (Sec. IV-E)

A. Dataset and Experimental Setup

This subsection presents two experimental datasets: the CPTC2018 dataset and our proposed **M**ulti-**S**tep **A**ttack **S**imulation (MSAS) dataset, along with system parameter configurations and quantitative metrics employed for comprehensive performance evaluation. Note that the CPTC2018 dataset comprises lightweight security logs (network alarms and application logs), while the MSAS dataset encompasses both lightweight security logs and audit logs. Table I summarizes the log statistics for both datasets.

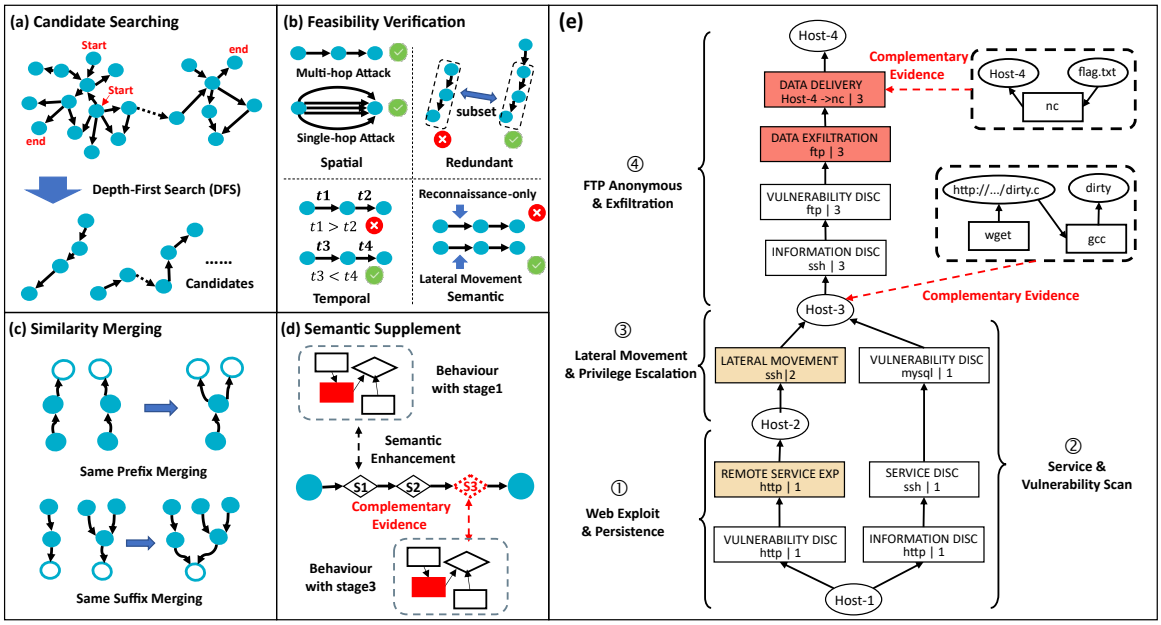

Fig. 8: Heuristic multi-step attack searching process. (a)-(d) depict four essential phases of the algorithm. (e) shows the reconstructed attack graph for the case presented in Fig. 2 of Sec. II, where ovals denote hosts, while rectangles depict events annotated with the attack stage, the predominantly targeted service, and the hop sequence number. The dashed region in the upper-right corner of (e) highlights complementary evidence identified in application logs.

TABLE I: Evaluation results of MuSAR across eight multi-step attack scenarios, presenting the log volume, IoC identification effectiveness with/without application logs, computational overhead, and semantic completion rate (SCR) of attack graphs.

Attack Scenario †	Count(10^3) ‡		Size(MB) ‡		Network-Only				Network & Host								Overhead		Attack Graph	SCR
	Light	Audit	Light	Audit	S-IoCs	TP	FN	Recall	S-IoCs	TP	FN	Recall	C-IoCs	TP	FN	Recall	Time	Memory		
T-1	109.4	-	112.3	-	192	189	3	98.4%	199	196	3	98.5%	1229	704	525	57.3%	5.02s	245M	46	86.96%
T-2	117.7	-	72.1	-	42	30	12	71.4%	46	44	2	95.7%	1132	766	366	67.7%	69.5s	231M	63	93.65%
T-5	73.6	-	71.1	-	128	113	15	88.3%	133	126	7	94.7%	644	450	194	69.9%	3.55s	226M	48	79.17%
T-7	79.9	-	66.2	-	132	123	9	93.2%	137	134	3	97.8%	852	560	292	65.7%	3.16s	225M	84	95.24%
T-8	86.7	-	76.9	-	140	126	14	89.9%	144	139	5	96.5%	1292	711	581	55.0%	3.87s	229M	63	85.71%
T-9	76.9	-	66.9	-	138	131	7	94.9%	142	137	5	96.5%	1566	798	768	51.0%	48.1s	230M	79	87.34%
Sum.	**544.2**	-	**465.5**	-	**772**	**712**	**60**		**801**	**776**	**25**		**6715**	**3981**	**2734**				**383**	
Avg.								**89.5%**				**96.6%**				**61.1%**	**22.2s**	**230M**		**88.01%**
S-1	3.3	18636	1.94	2905	142	141	1	99.3%	149	148	1	99.3%	30	15	15	50.0%	0.40s	187M	4	75.00%
S-2	2.1	25254	0.51	3298	68	68	0	100%	68	68	0	100%	5	2	3	40.0%	0.34s	187M	14	64.29%
Sum.	**2.7**	**43890**	**2.45**	**6203**	**210**	**209**	**1**		**217**	**216**	**1**		**35**	**17**	**18**				**18**	
Avg.								**99.7%**				**99.7%**				**45.0%**	**0.37s**	**187M**	**9**	**69.65%**

† Scenarios T-1 to T-9 come from the CPTC2018 dataset, while Scenarios S-1 and S-2 are from the MSAS dataset.
‡ "Light" refers to lightweight security logs, including network alarms and application logs, while "Audit" denotes audit logs. Due to table length restrictions, a more detailed overview of log count and size is provided in Appendix B1.

1) CPTC2018 Dataset: The CPTC2018 dataset [14] records a nine-hour penetration testing exercise conducted by six teams (T-1, T-2, T-5, T-7, T-8, T-9) in an automotive enterprise network. Each team operated in an isolated environment, effectively creating distinct multi-step attack scenarios. The dataset encompasses over 330,000 network alarms alongside various types of application logs, all presented without ground truth annotations. For our analysis, we label network alarms where both the attacker and victim are Local Area Network (LAN) addresses (i.e., `10.0.*.*`) and all historical command logs (i.e., `.bash_history`) as attack-related traces.

2) MSAS Dataset: Based on attack techniques from CPTC2018, we replicated two distinct multi-step attack scenarios in a controlled testbed with crafted vulnerabilities to create a labeled dataset for rigorous evaluation. As illustrated in Fig. 9, the testbed consists of three zones: the internet, the DeMilitarized Zone (DMZ), and the internal network. All hosts are virtual machines, with the attacker operating Kali Linux and other hosts running Ubuntu. Log collection frameworks, including Sysdig [31], Suricata [17], and the Elastic Stack [32], are deployed in the DMZ and internal network zones to capture lightweight security logs (network alarms and application logs), and audit logs.

Attack Scenarios. In scenario 1 (S-1), attackers gained initial access from the internet, laterally moved within the internal network, and ultimately exploited FTP anonymous

TABLE II: Attack step reconstruction analysis for the MSAS dataset, detailing attack techniques, presence of attack-related traces in network alarms and application logs, reconstruction results, and recall of IoC identification for each attack step.

Attack Steps		Step-1	Step-2	Step-3	Step-4	Step-5	Step-6	Step-7	Step-8	Step-9	Step-10	**Sum./Avg.**
S-1	Techniques †	**RE**	**EXP+IA**	**HE**	**EXP+PR**	**LM**	**PE**	**RE**	**CA**	**LM**	**DE**	-
	Network	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	**8**
	Application	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	**7**
	Recover	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	**9**
	Recall	100.00%	100.00%	-	100.00%	55.56%	55.56%	100.00%	100.00%	46.15%	87.50%	**91.06% (163/179)**
S-2	Techniques †	**IA**	**PE**	**RE**	**HE**	**LM**	**EV**	**PR**	**EXP**	**CA**	**DE**	-
	Network	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	**8**
	Application	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	**2**
	Recover	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	**9**
	Recall	57.14%	100.00%	100.00%	-	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	**95.89% (70/73)**

† RE: Reconnaissance. IA: Initial Access. EXP: Vulnerability Exploitation. PE: Persistence. EV: Defense Evasion. PR: Privilege Escalation. CA: Credential Access. LM: Lateral Movement. HE: Host Exploration. DE: Data Exfiltration.

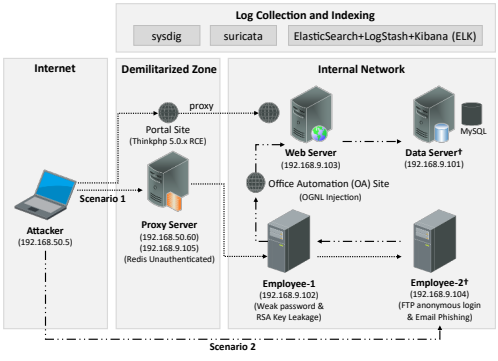

Fig. 9: The network architecture and predefined vulnerability settings of the controlled testbed. † indicates the ultimate targets in both attack scenarios.

login to exfiltrate data from employee-2, as depicted by the dotted path in Fig. 9. In scenario 2 (S-2), attackers established an initial foothold through email phishing and then exploited vulnerabilities in the OA site to steal sensitive data from the data server, shown by the dash-dotted path in Fig. 9.

Labeling Ground Truth. We collected a total of 5,369 lightweight logs (∼2.5MB), as well as nearly 44 million audit logs (∼6.2GB), as shown in Table I. Based on the ground truth, we partitioned each scenario into ten discrete steps, denoted as step-1 ~ step-10 (see Appendix B2), and annotated each attack-related trace with a label of its corresponding attack step. For example, in S-1, a network alarm with the signature `ET FTP FTP PWD command attempt without login` is labeled as step-10, which represents data exfiltration from employee-2. Table II shows the techniques employed in each step. More details about the distribution of attack-related traces for both scenarios can be found in Appendix B3.

Note that as a typical pentest scenario, attackers in the CPTC2018 dataset follow typical penetration testing patterns, prioritizing speed over stealth to capture flags. To cover the non-pentest scenario, the MSAS dataset incorporates covert attack techniques designed to evade security tools, resulting in some unobservable attack steps, e.g., Step-3 in S-1 and Step-4 in S-2. Together, these datasets enable comprehensive evaluation against both overt, verifiable attacks and stealthy techniques.

3) Parameters: Similar to SAGE, we configure the window parameter to 150 seconds ($wLen = 150$) for extracting inter-host abnormal episodes. For intra-host sensitive behavior aggregation, we empirically determine the weight parameters through manual analysis, with the sequential weight, similarity weight and association weight set to 5, 3 and 1, respectively. We choose the Qwen-turbo [29] model to establish relationships between intra-host sensitive behaviors and corresponding attack stages. All experiments are conducted on a server equipped with an Intel Xeon 6266C processor and 1TB of RAM.

4) Evaluation Metrics: The effectiveness of attack reconstruction hinges on two key capabilities: identifying security events that indicate attack actions from attack-related traces, and accurately restoring relationships between these events. To systematically evaluate this process, we define two types of **I**ndicators **o**f **C**ompromise (IoCs): **S**ignature-based **IoCs** (**S-IoCs**) for inter-host connections and **C**ommand-based **IoCs** (**C-IoCs**) for intra-host operations. These IoCs represent the core semantic elements of abnormal episodes and sensitive behaviors that reflect attackers' activities. We evaluate MuSAR's reconstruction accuracy by comparing identified IoCs against ground-truth labels using standard metrics: recall and F1-score. An attack step i is considered successfully reconstructed when MuSAR correctly identifies its corresponding IoC. Furthermore, to quantify the enhancement in attack graph completeness by integrating application logs, we define the **S**emantic **C**ompletion **R**ate (**SCR**) metric, which measures the proportion of attack graphs that can be augmented with supplementary evidence from application logs when compared to using network alarms alone.

B. RQ1: Effectiveness of MuSAR

Overall Effectiveness. Table I presents MuSAR's reconstruction performance across eight multi-step attack scenarios. For the CPTC2018 dataset, MuSAR compresses approximately 544.2k lightweight log entries into 383 attack graphs, with an average time of 22.2s and memory consumption of 230MB. The analysis reveals that 88.01% of the generated

graphs incorporate evidence that would be missing when using network alarms exclusively. For the MSAS dataset, MuSAR efficiently processes 5,369 log entries into 18 attack graphs within one second, achieving SCRs of 75% and 64% for the two scenarios, respectively. The results demonstrate MuSAR's ability to perform semantic associations between lightweight security logs and achieve an impressive compression rate of approximately three orders of magnitude. More detailed results during the reconstruction are provided in Appendix B1.

Identification of IoCs. As shown in Table I, we compare MuSAR's IoC identification performance with and without application logs. From the table, incorporating application logs increases the average recall of S-IoCs on the CPTC2018 dataset from 89.5% to 96.6%. This improvement indicates that MuSAR effectively extracts malicious connections from the application logs and associates previously disconnected network alarms, enabling more comprehensive identification compared to network-only analysis. In terms of C-IoCs, MuSAR identified 61.1% and 45.0% of sensitive operations from the CPTC2018 and MSAS datasets, respectively, to construct intra-host sensitive behaviors. Analysis reveals that file and directory manipulation commands (e.g., `cd`, `ls`, and `cat`) constitute the majority of false negatives across all eight attack scenarios (see Fig. 14 in Appendix B1 for more details). These misclassified commands are not integral to the core semantics of attack actions. Therefore, their missing from MuSAR's analysis does not significantly impact the attack reconstruction results.

Attack Step Reconstruction. Given the limited descriptive information in the CPTC2018 dataset, we conduct the evaluation of attack step reconstruction on the MSAS dataset. As shown in Table II, although individual log types fail to capture all attack steps—particularly in S-2, where only two steps (step-1 and step-5) were recorded in application logs—MuSAR successfully reconstructs 9 out of 10 steps for both scenarios, mitigating the limitations imposed by single-type logs. From the table, MuSAR achieves an average recall of 91.06% (163 out of 179) and 95.89% (70 out of 73) for both scenarios, respectively. Among the 19 false negatives, 18 are C-IoCs associated with directory and file operations. The remaining S-IoC, designated as `POSSBL SCAN SHELL M-SPLOIT TCP`, originates from remote control traffic between the attacker and the proxy server, which is disregarded by MuSAR due to its isolation from others and the fact that it involves only one single alarm signature. However, MuSAR successfully reconstructs the attack action from the proxy server's outbound traffic, thereby enabling the perception of external malicious hosts despite this omission.

C. RQ2: Explaining Attack Strategies

We illustrate MuSAR's analytical capabilities through a four-hop attack reconstructed from S-2, as shown in Fig. 10. This graph condenses 450 log entries into 14 event nodes. MuSAR accurately reconstructs the multi-step attack, achieving 100% recall for IoC identification. According to the ground truth (see Appendix B2), attackers initially compromised host-104 via email phishing, then performed lateral movement to host-102 by exploiting an RSA key leakage. Subsequently, they gained control over host-103 via a vulnerability in the OA site and finally exfiltrated data from host-101.

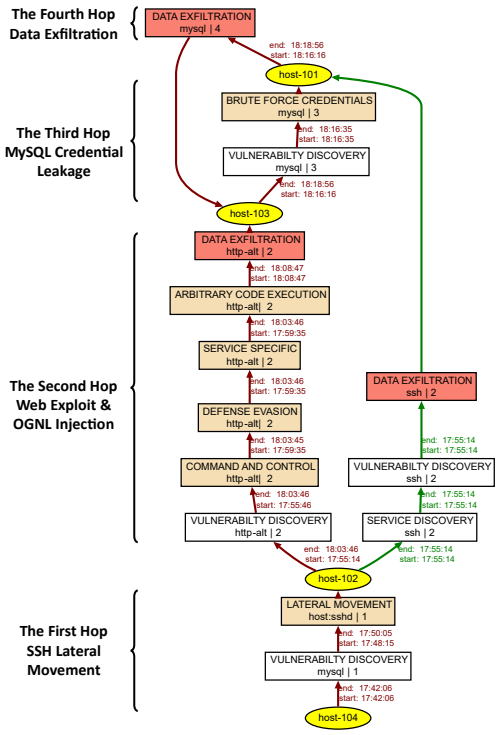

Fig. 10: A four-hop attack reconstructed by MuSAR from S-2. Ovals and rectangles represent hosts and events, respectively. Each event node comprises three elements: attack stage, predominantly targeted service, and hop sequence number. The color intensity indicates the severity level of events. Arrow colors differentiate parallel paths within each hop. Label 'host-xxx' represents the host with IP address `192.168.9.xxx`.

(1) In the first hop (from host-104 to host-102), network alarms only capture the [VULNERABILITY DISCOVERY] action. By incorporating application logs, MuSAR effectively identifies potential malicious connections and reveals an additional episode targeting host-102, involving [LATERAL MOVEMENT] stage and `host:sshd` service, which provides critical evidence of the exploitation of RSA key leakage.

(2) In the second hop (from host-102 to host-103 and host-101), MuSAR reconstructs parallel attack paths targeting host-101 (green arrows) and host-103 (maroon arrows), respectively. Although both paths initiate simultaneously, the subsequent events exhibit a sequential execution pattern. Specifically, the second event targeting host-103 ([COMMAND AND CONTROL] node) starts at `17:55:46`, which follows the completion of the final event ([DATA EXFILTRATION] node) targeting host-101 at `17:55:14`. This temporal pattern indicates that after initial vulnerability scanning, attackers pivoted to exploit vulnerable host-103. Analysis of alarm

TABLE III: Comparison of MuSAR with SOTA approaches on the MSAS dataset.

Scenarios & Methods		Missing Steps †	Cost		Recall	F1-score
			Time	Mem.		
S-1	SAGE	{3,4,5}	0.51s	**138M**	77.78%	83.33%
	ATLAS	{2,5,9}	6695s	12.6G	23.88%	21.62%
	MAGIC	{1,2,3,4,5,9,10}	186s	327M	5.26%	10.00%
	MuSAR	**{3}**	**0.40s**	187M	**91.06%**	**92.88%**
S-2	SAGE	{1,2,3,4,5,6,10}	0.44s	**138M**	60.27%	71.54%
	ATLAS	{4,6,8,10}	6820s	12.5G	18.09%	30.63%
	MAGIC	{3,4,5,7,8,9,10}	195s	323M	2.94%	5.71%
	MuSAR	**{4}**	**0.34s**	187M	**95.89%**	**95.89%**

† We use 1 ~ 10 to represent step-1 through step-10.

signatures reveals attackers leveraged an OGNL expression injection vulnerability in the OA site.

(3) In the third hop (from host-103 to host-101), MuSAR reveals vulnerability scanning and events indicating credential cracking attempts against the MySQL service on host-101, confirming the successful compromise of host-103.

(4) In the fourth hop (from host-101 to host-103), MuSAR reconstructs MySQL database authentication followed by data exfiltration activities. Integration of attack semantics across four hops reveals the complete attack progression, enabling comprehensive analysis of adversarial strategies and tactics.

D. RQ3: Comparison Analysis

We evaluate MuSAR against representative SOTA methods from two dominant paradigms in attack reconstruction. First, we evaluate against SAGE [5], a state-based heuristic model that reconstructs attacks from network alarms. Second, we compare MuSAR with provenance-based approaches, which are learning-based models that leverage audit logs for high-fidelity reconstruction. All learning-based models are retrained on our MSAS dataset for fair comparison. The results show that MuSAR achieves comparable or superior reconstruction accuracy in complex, multi-host scenarios while maintaining better scalability and requiring only lightweight data sources. The details are as follows.

MuSAR vs. SAGE. As shown in Table III, evaluation on the MSAS dataset demonstrates that MuSAR achieves comparable time cost (<0.5s) and memory usage (<200MB) to SAGE [5] while outperforming it in both recall and F1-score. In S-1, SAGE demonstrates effective reconstruction with a recall of 77.78% and an F1-score of 83.33%. However, its performance degrades in S-2, reconstructing only three attack steps with a 60.27% recall and 71.54% F1-score. This performance decline stems from two critical factors: (1) SAGE exclusively relies on network alarms and disregards low-frequency states during the model's learning phase unless they exhibit high risks, and (2) S-2 involves multiple stealthy techniques, such as lateral movement via SSH keys and communication through SOCKS5 tunnels, which typically evade IDS detection or generate only low-risk alarms. These constraints limit SAGE's effectiveness in attack scenarios involving incomplete network alarms and sophisticated attack patterns. We also compare attack graphs generated for identical multi-step attacks on the CPTC2018

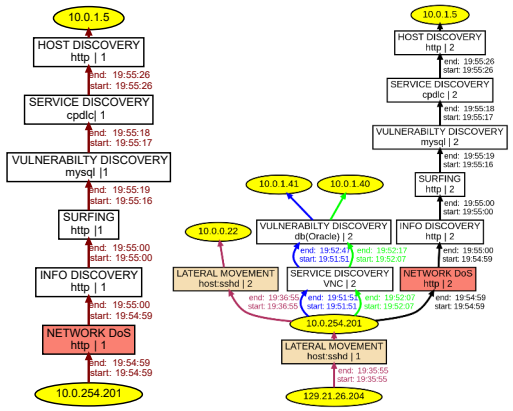

Fig. 11: Comparison of attack reconstruction between SAGE (left) and MuSAR (right) for an identical multi-step attack in T-8, featuring a shared attack path from `10.0.254.201` to `10.0.1.5`.

dataset. As shown in Fig. 11, compared to SAGE, MuSAR identifies two additional episodes (i.e., two wheat-colored nodes in the figure) from application logs, revealing potential SSH-based lateral movement activities that are undetectable in network alarms. By incorporating application logs, MuSAR associates four additional risk hosts, enabling a more comprehensive and precise attack reconstruction.

MuSAR vs. Provenance-based Approaches. Analysis reveals that audit logs in the MSAS dataset capture attack-related traces for all attack steps, including step-3 in S-1 and step-4 in S-2, which are undetectable in both network alarms and application logs. However, provenance-based approaches, including ATLAS [15] and MAGIC [16], fail to achieve satisfactory performance with a limited number of attack steps identified, resulting in relatively poor recall and F1-score, as shown in Table III. The performance discrepancy can be attributed to two reasons. First, existing provenance-based methods are primarily designed for single-host environments and have difficulty modeling cross-host attack propagation in the MSAS dataset. Second, these methods focus on modeling interactions between malicious entities (e.g., I/O operations and process spawning) while overlooking critical semantic indicators (e.g., anomalous arguments). However, the MSAS dataset involves multiple attacks conducted through legitimate processes (e.g., `sh`), making it challenging for ATLAS and MAGIC to reconstruct attack steps due to the scarcity of clearly identifiable malicious entities. MuSAR focuses on the semantic relationships between underlying actions behind entities, distinguishing between legitimate and malicious activities through complementary perspectives (e.g., SSH-based lateral movement, typically undetectable in network alarms, exhibits distinctive anomalies in application logs), facilitating an effective attack reconstruction in multi-host environments. Furthermore, in contrast to MuSAR, substantial time and memory costs presented in Table III limit the deployment of provenance-based approaches in real-world environments.

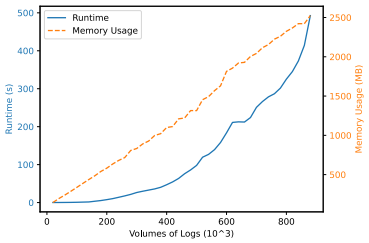

Fig. 12: Runtime and memory usage vs. log volume using real-world datasets in production environments.

E. RQ4: Feasibility of Deployment

Deployment Complexity. MuSAR seamlessly integrates with existing security infrastructures with minimal setup requirements. It directly ingests network alarms from security devices and utilizes application logs without requiring specialized configurations. Only basic log collection agents (e.g., Filebeat [33]) are required for centralized log aggregation.

Analysis of Long-Term Offline and Real-time Reconstruction. We have analyzed security logs from three large energy enterprises, each running with thousands of hosts. These enterprises suffer from billions of network connections and thousands of application logs per month, generating approximately three million lightweight security logs. After eliminating false positive alarms, there are still around 500k logs remaining for further investigation. Leveraging these datasets (restricted due to security considerations), we evaluate MuSAR's feasibility in attack reconstruction. As shown in Fig. 12, memory usage scales linearly with the log volume. Although the runtime exhibits approximately exponential growth, it is still acceptable in both long-term offline and real-time attack reconstruction. Specifically, for monthly-scale analysis ($\sim$500k logs), MuSAR completes processing within a few minutes, which is much faster than provenance-based approaches (e.g., the substantial runtime of ATLAS and MAGIC, as shown in Table III). For routine monitoring at hourly and daily scales ($<$100k logs), MuSAR responds in seconds, enabling effective interactive analysis. Furthermore, we develop an interactive real-time analysis system that enables analysts to rapidly identify and trace multi-step attacks, facilitating efficient threat detection and response. A demo system integrating core functionalities of MuSAR is available at https://bit.ly/4h11wwZ.

V. Discussion

Scalability and Transferability. Unlike previous approaches that construct associations between logs [23], [34], MuSAR provides a framework for log-level structured representation and event-level semantic alignment. Only customized parsing and attack stage mapping rules are required for log integration. (The process of formulating rules is detailed in Appendix E.) Furthermore, as discussed in Sec. IV-E, MuSAR integrates seamlessly with existing security infrastructures, ensuring operational scalability and practical transferability.

Complementarity between MuSAR and Provenance-Based Approaches. Provenance-based methods detect anomalies within a single host using audit logs, but struggle with large-scale cross-host attacks due to high computational and storage costs. In contrast, MuSAR efficiently associates data across multiple log sources with low runtime overhead. This complementary nature suggests an integrated approach: deploying provenance-based monitoring on critical hosts to supplement lightweight security logs, while leveraging MuSAR's efficient correlation capabilities. Such integration would enhance both the precision and coverage of attack reconstruction in multi-host environments, effectively balancing detection capabilities with system overhead.

False Positives. False alarms generated by security tools pose significant challenges in real-world environments [35]. These alerts, often triggered by benign activities, are intermingled with attack-related logs, increasing the prevalence of false positives (FPs) in noisy scenarios. Specifically, MuSAR may construct spurious multi-step attack graphs due to erroneous correlations among unrelated events. Such FPs typically arise from: (1) concurrent benign operations, where multiple administrators or automated tools perform maintenance or security scans on overlapping hosts within a short period, inadvertently mimicking the spatial and temporal patterns of attacks; and (2) high-density, low-severity events, where a large volume of low-priority alerts in extensive networks statistically increases the likelihood of random events forming seemingly logical sequences. To address this issue, we can utilize alert triage [36] and prioritization [37] techniques to filter out the majority of false alarms before heterogeneous event generation in MuSAR. Furthermore, we have incorporated a multi-dimensional feasibility verification (FV) process (Sec. III-C2) into MuSAR. As demonstrated in Appendix C, the FV module is highly effective at reducing false positives while preserving detection accuracy.

False Negatives and Adversarial Settings. False negatives (FNs) primarily occur under three conditions. First, as detailed in Appendix B1, FNs emerge from the intentional exclusion of low-level auxiliary commands (e.g., `cd`, `ls`, `cat`) that lack substantive attack semantics. While this filtering strategy contributes to FNs, it produces more concise attack graphs that highlight the adversary's strategic actions rather than operational minutiae. Second, FNs occur when MuSAR filters isolated alarms lacking corroborating evidence. For example, as observed in Table I, an initial `POSSBL SCAN SHELL M-SPLOIT TCP` alarm for S-1 has been discarded. Despite pruning this low-confidence alarm, MuSAR can successfully reconstruct the subsequent attack sequence using higher-confidence evidence from the proxy server's outbound traffic. Third (and most critically), FNs are unavoidable when attack-related artifacts are incomplete, whether due to evasion techniques that bypass detection or deliberate log manipulation that disrupts causal chains. Specifically, MuSAR fails to detect step-3 in S-1 and step-4 in S-2 (see Table II) due to insufficient log coverage, fundamentally constraining its detection capabilities to the observability scope of the lightweight security logs.

In adversarial settings, FNs can arise when the log integrity is compromised. Adversaries with sufficient privileges may disrupt the causal chain by tampering with or erasing logs, resulting in incomplete attack reconstructions. This vulnerability can be mitigated by integrating MuSAR with robust log integrity verification methods [19]–[21].

Aided Forensic Analysis for Zero-day Attacks. While zero-day attacks can circumvent traditional defenses by exploiting unknown vulnerabilities [38], their post-exploitation activities for the final attack goal (e.g., establishing control authority or escalating privileges) invariably leave forensic evidence on compromised hosts that MuSAR can identify. For example, in Scenario S-1 of the MSAS dataset (see Table III), multiple SOTA detection systems (SAGE, ATLAS, and MAGIC) failed to identify step 5: a lateral movement phase. MuSAR, however, successfully reconstructs this attack stage by semantically correlating disparate application log events. Specifically, MuSAR identified the attack progression by linking a proxy server authentication event with subsequent reconnaissance activities (e.g., `whoami`, `ping` commands). Thus, when the initial exploit evades detection by zero-day attacks, MuSAR's semantic correlation capabilities can help reveal crucial forensic evidence of post-exploitation activities.

Future Work. Like other attack reconstruction frameworks [7], [15], MuSAR's primary limitation lies in its dependence on log availability and integrity. In the future, we aim to enhance MuSAR's operational resilience and analytical precision through several key initiatives: (1) advancing causal inference techniques to reconstruct missing attack stages in multi-step attack sequences; (2) deploying and evaluating our framework in large-scale enterprise environments to assess its forensic capabilities against previously unknown attacks (e.g., zero-day attacks); (3) expanding the system's coverage by incorporating additional log sources and developing a hybrid analysis framework that combines our approach with state-of-the-art provenance-based methods; and (4) conducting systematic quantitative analysis of our heuristic rules to evaluate their maintenance overhead and effectiveness, thus validating the framework's practical utility.

VI. Related Work

Semantic Association for Security Events. Early research explored relationships between security logs [34], [39], [40], exemplified by HERCULE [34], which uses 29 association rules across six types of logs to construct a multi-dimensional weighted graph to represent host relationships. Unfortunately, these approaches fail to capture underlying attack semantics, underutilizing the rich contextual information available in logs. Recent research has proposed various semantic association methods to understand attack actions. Initial efforts primarily focused on single log types, such as network alarms [5], [6], or audit logs [7], [8], [10]. However, the evolution of attack techniques have demonstrated the limitations of single-source analysis [4], prompting increased attention toward the integration of diverse logs [41]–[43], and high-level semantic events, such as cyber threat intelligence [9], [44] and user-level click events [45], [46]. Furthermore, inspired by the similar structure between log text and natural language, many studies have employed deep learning techniques for semantic association [42], [47]–[50]. However, these methods struggle to effectively bridge the semantic gap between heterogeneous events, often requiring extensive manual analysis. In contrast, MuSAR achieves event-level semantic alignment within a unified attack lifecycle, accurately identifying relationships between events and attack stages.

Attack Investigation and Reconstruction. Attack investigation and reconstruction serve as crucial components in threat response, enabling analysts to identify compromised hosts and system vulnerabilities. Modern approaches predominantly rely on provenance-based methods, which leverage system entity causality in audit logs to reconstruct attack paths [8], [15], [40], [51]–[55]. For instance, DEPCOMM [8] summarizes provenance graphs by partitioning them into process-centric communities and restores attack paths through information flow analysis to capture attack-related processes. ATLAS [15] constructs an end-to-end attack story by integrating causality analysis, natural language processing, and sequence learning techniques to identify attack-related events within the provenance graph. However, these methods largely depend on the prior identification of malicious entities. In practice, attackers often adapt their tactics to obscure their traces (e.g., processes and files) and evade similarity-based detection mechanisms [56]. Furthermore, provenance-based methods typically suffer from substantial overhead, limiting their widespread deployment in production environments [13]. In contrast, MuSAR can identify fragmented traces from lightweight security logs without requiring any prior attack-related information and effectively implement semantic association to reconstruct potential multi-step attacks with minimal effort.

VII. Conclusion

We present MuSAR, a framework that analyzes lightweight security logs using log-level structured representations and event-level semantic associations. MuSAR can reconstruct potential multi-step attacks without requiring attack-related ground truth in multi-host environments. Evaluation on two distinct multi-step attack datasets demonstrates that MuSAR effectively identifies most attack-related traces with minimal effort and reconstructs multi-step attacks across multiple hosts with strong interpretability. These capabilities provide security analysts with actionable insights, significantly enhancing incident response against sophisticated attacks.

Acknowledgment

The authors thank the anonymous shepherd and reviewers for their comments and suggestions. This work was partially supported by National Natural Science Foundation of China (62293500, 62293501, 62293502) and Fundamental Research Funds for the Central Universities.

REFERENCES

[1] X. Wang, X. Gong, X. Zhang, and Z. Cheng, "A survey of multi-step attack detection," *Journal of Cyber Security*, 2021.

[2] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

[3] J. Navarro, A. Deruyver, and P. Parrend, "A systematic survey on multi-step attack detection," *Computers & Security*, vol. 76, pp. 214–249, 2018.

[4] C. Xosanavongsa, E. Totel, and O. Bettan, "Discovering correlations: A formal definition of causal dependency among heterogeneous events," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 340–355.

[5] A. Nadeem, S. Verwer, S. Moskal, and S. J. Yang, "Alert-driven attack graph generation using S-PDFA," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 731–746, 2021.

[6] S. Haas and M. Fischer, "On the alert correlation process for the detection of multi-step attacks and a graph-based realization," *ACM SIGAPP Applied Computing Review*, vol. 19, no. 1, pp. 5–19, 2019.

[7] X. Yang, H. Liu, Z. Wang, and P. Gao, "Zebra: Deeply integrating system-level provenance search and tracking for efficient attack investigation," *arXiv preprint arXiv:2211.05403*, 2022.

[8] 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)*. IEEE, 2022, pp. 540–557.

[9] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. Venkatakrishnan, "Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 1795–1812.

[10] 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)*, 2023, pp. 6575–6592.

[11] M. N. Hossain, J. Wang, O. Weisse, R. Sekar, D. Genkin, B. He, S. D. Stoller, G. Fang, F. Piessens, E. Downing *et al.*, "Dependence-preserving data compaction for scalable forensic analysis," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1723–1740.

[12] 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*, 2018, pp. 1324–1337.

[13] M. Zipperle, F. Gottwalt, E. Chang, and T. Dillon, "Provenance-based intrusion detection systems: A survey," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–36, 2022.

[14] N. Munaiah, J. Pelletier, S.-H. Su, S. Yang, and A. Meneely, "A cybersecurity dataset derived from the national collegiate penetration testing competition," in *HICSS Symposium on Cybersecurity Big Data Analytics*, 2019.

[15] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "ATLAS: A sequence-based learning approach for attack investigation," in *30th USENIX security symposium (USENIX security 21)*, 2021, pp. 3005–3022.

[16] 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.

[17] "Suricata," https://suricata.io/.

[18] "MITRE ATT&CK®," https://attack.mitre.org/.

[19] M. Montanari, J. H. Huh, D. Dagit, R. B. Bobba, and R. H. Campbell, "Evidence of log integrity in policy-based security monitoring," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*, 2012, pp. 1–6.

[20] B. Putz, F. Menges, and G. Pernul, "A secure and auditable logging infrastructure based on a permissioned blockchain," *Computers & Security*, vol. 87, p. 101602, 2019.

[21] J. Chen, X. Chen, K. He, R. Du, W. Chen, and Y. Xiang, "DELIA: Distributed efficient log integrity audit based on hierarchal multi-party state channel," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3286–3300, 2021.

[22] S. Moskal, S. J. Yang, and M. E. Kuhl, "Extracting and evaluating similar and unique cyber attack strategies from intrusion alerts," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2018, pp. 49–54.

[23] X. Wang, X. Yang, X. Liang, X. Zhang, W. Zhang, and X. Gong, "Combating alert fatigue with AlertPro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection," *Computers & Security*, vol. 137, p. 103583, 2024.

[24] IANA, "Service name and transport protocol port number registry," https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, 2021.

[25] S. Moskal and S. J. Yang, "Framework to describe intentions of a cyber attack action," *arXiv preprint arXiv:2002.07838*, 2020.

[26] "Wazuh - open source xdr. open source siem." https://wazuh.com/.

[27] "Idank/bashlex: Python parser for bash," https://github.com/idank/bashlex.

[28] "Uptycs linux commands & utilities commonly used by attackers," https://www.uptycs.com/blog/threat-research-report-team/linux-commands-and-utilities-commonly-used-by-attackers.

[29] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[30] "Graphviz," https://graphviz.org/.

[31] "Sysdig | security for containers, kubernetes, and cloud," https://sysdig.com/.

[32] "Elastic stack: (ELK) elasticsearch, kibana & logstash," https://www.elastic.co/elastic-stack.

[33] "Filebeat: Lightweight log analysis & elasticsearch | elastic," https://www.elastic.co/beats/filebeat.

[34] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack story reconstruction via community discovery on correlated log graph," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, 2016, pp. 583–595.

[35] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of SOC analysts' perspectives on security alarms," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2783–2800.

[36] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian, "Human-guided machine learning for fast and accurate network alarm triage," in *IJCAI'11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. AAAI, July 2011, pp. 2564–2569, invited from CHI 2011 to Best Papers Track.

[37] Y. Liu, G. Ruan, Z. Luo, S. Zhang, D. Liu, X. Fan, Y. Zhou, and T. Liu, "DeepDRAC: Disposition recommendation for alert clusters based on security event patterns," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 6443–6458, 2025.

[38] R. Ahmad, I. Alsmadi, W. Alhamdani, and L. Tawalbeh, "Zero-day attack detection: a systematic literature review," *Artificial Intelligence Review*, vol. 56, no. 10, pp. 10 733–10 811, 2023.

[39] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, and M. Rajarajan, "Intrusion alert prioritisation and attack detection using post-correlation analysis," *Computers & security*, vol. 50, pp. 1–15, 2015.

[40] J. Liu, J. Yan, Z. Jiang, X. Wang, and J. Jiang, "A graph learning approach with audit records for advanced attack investigation," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 897–902.

[41] S. Haas, R. Sommer, and M. Fischer, "Zeek-osquery: Host-network correlation for advanced monitoring and intrusion detection," in *ICT Systems Security and Privacy Protection*. Cham: Springer International Publishing, 2020, pp. 248–262.

[42] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[43] W. U. Hassan, M. A. Noureddine, P. Datta, and A. Bates, "OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis," in *Network and Distributed System Security Symposium*, 2020.

[44] X. Zang, J. Gong, X. Zhang, and G. Li, "Attack scenario reconstruction via fusing heterogeneous threat intelligence," *Computers & Security*, vol. 133, p. 103420, 2023.

[45] R. Yang, S. Ma, H. Xu, X. Zhang, and Y. Chen, "UIScope: Accurate, instrumentation-free, and visible attack investigation for GUI applications." in *NDSS*, 2020.

[46] S. Song, X. Liu, X. Fu, B. Luo, X. Du, and M. Guizani, "Visible forensic investigation for android applications by using attack scenario reconstruction," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.
[47] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 807–817.
[48] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1196–1201.
[49] R. Chen, S. Zhang, D. Li, Y. Zhang, F. Guo, W. Meng, D. Pei, Y. Zhang, X. Chen, and Y. Liu, "Logtransfer: Cross-system log anomaly detection for software systems with transfer learning," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 37–47.
[50] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "Semi-supervised log-based anomaly detection via probabilistic label estimation," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1448–1460.
[51] 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 *NDSS*, 2021.
[52] 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.
[53] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *arXiv preprint arXiv:2001.01525*, 2020.
[54] H. Irshad, G. Ciocarlie, A. Gehani, V. Yegneswaran, K. H. Lee, J. Patel, S. Jha, Y. Kwon, D. Xu, and X. Zhang, "TRACE: Enterprise-wide provenance tracking for real-time apt detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4363–4376, 2021.
[55] J. Zengy, 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)*. IEEE, 2022, pp. 489–506.
[56] A. Goyal, X. Han, G. Wang, and A. Bates, "Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems," in *30th Network and Distributed System Security Symposium*, 2023.

APPENDIX

A. Details of Security Event Aggregation

1) Anomalous Log Patterns from Application Logs: In this paper, five types of representative lightweight security logs from Linux platforms are consolidated into two types of log-level traces: inter-host connections and intra-host operations. Network alarms, which capture anomalous connections between hosts, constitute a primary component of inter-host connections, while historical command log (i.e., `.bash_history`) serves as a critical resource of intra-host operations. The remaining three categories of logs (`auth.log`, `imap.log` and `mongo.log`) record running status and comprehensive communications on specific applications, enabling the detection of inter-host connections that network alarms might miss. Based on Wazuh, we identify anomalous log patterns and enhance the detection rules to extract potential inter-host connections, as detailed in Table IV.

2) Classification Criteria Between Linux Commands and Attack Stages: Table V showcases representative examples of the extended classification criteria between sensitive entities (including processes like `nmap`, files like `authorized_keys`, and patterns like `-i>& /dev/tcp`), and their corresponding attack stages within the ATT&CK framework.

3) Event-level Semantic Alignment under MITRE ATT&CK framework: Our approach aligns both inter-host abnormal episodes and intra-host sensitive behaviors with attack stages. These stages are defined by 14 adversarial tactics in the ATT&CK framework, which provides a comprehensive knowledge base of adversarial techniques.

Rule-based Semantic Alignment of Inter-host Episodes. Inter-host abnormal episodes are consolidated sequences of inter-host connections that characterize related security events between communicating hosts. These episodes are initially classified using alarm signatures based on the Action-Intent-Framework (AIF). To map these episodes to the ATT&CK framework's 14 attack stages, we implement explicit rules correlating 35 AIF techniques with corresponding ATT&CK tactics, as detailed in Table VI.

LLM-based Semantic Alignment of Intra-host Behaviors. Intra-host sensitive behaviors, primarily derived from historical command logs, are causally associated with intra-host operations that characterize security events occurring within a host. Given the diversity and context-dependent nature of command logs, direct rule-based mapping of these behaviors to attack stages is impractical. Instead, as illustrated in Figure 13, we implement a two-phase LLM-based semantic alignment method that transforms stage mapping from a broad search problem into a precise matching task:

(a) Phase 1: Stage Candidate Generation. Initially, the LLM analyzes the command sequence against descriptions of all 14 attack stages, performing coarse-grained classification to identify the three most probable stage candidates. This filtering step reduces the mapping scope from 14 to 3 stages.

(b) Phase 2: Final Stage Determination. Subsequently, we employ in-context learning for precise stage selection. For each candidate stage identified in Phase 1, we retrieve relevant command examples from an expert-annotated knowledge base. Using this focused context, the LLM determines the best-matching stage based on historical examples.

To mitigate LLM hallucinations that could produce inconsistent or verbose outputs, we implement a structured response mechanism requiring the model to encapsulate its final stage classification within `<begin>` and `<end>` tags. This design enables precise extraction of stage labels while filtering extraneous content. Additionally, the system includes automatic validation: if the extracted label is invalid, the query is re-executed, ensuring output reliability.

B. Dataset Details

1) Details of Eight Scenarios on Two Datasets: Table VII summarizes eight multi-step attack scenarios from CPTC2018 and MSAS dataset, detailing:

- Log statistics (count and size) for network alarms, application logs, and audit logs.
- Log attributes (IP addresses, services, signatures) and scenario duration.

TABLE IV: Anomalous log patterns and corresponding attributes for extracting inter-host connections from application logs.

Source	Log Pattern	Signature †	Severity	Attack Stage †
auth.log	**<time> <host>** sshd[**<pid>**]: PAM 1 more authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=**<sip>** user=**<user>**	Authentication failure	Medium	PRIV_ESC
	<time> <host> sshd[**<pid>**]: error: Received disconnect from **<sip>** port **<sport>**:3: com.jcraft.jsch.JSchException: Auth fail [preauth]	Invalid user login attempts	Medium	BRUTE_FORCE
	<time> <host> sshd[**<pid>**]: Failed password for invalid user **<user>** from **<sip>** port **<sport>** ssh2	Invalid user login attempts	Medium	BRUTE_FORCE
	<time> <host> sshd[**<pid>**]: Failed none for invalid user **<user>** from **<sip>** port **<sport>** ssh2	Invalid user login attempts	Medium	BRUTE_FORCE
	<time> <host> sshd[**<pid>**]: invalid user **<user>** from **<sip>** port **<sport>** ssh2	Invalid user login attempts	Medium	BRUTE_FORCE
	<time> <host> sshd[**<pid>**]: Accepted password for **<user>** from **<sip>** port **<sport>** ssh2	remote login successfully	High	LATERAL
	<time> <host> sshd[**<pid>**]: Accepted publickey for **<user>** from **<sip>** port **<sport>** ssh2: RSA <key>	remote login successfully	High	LATERAL
imap.log	**<time> <host>** dovecot: imap-login: Disconnected: Too many invalid commands (no auth attempts in 0 secs): user=**<user>**, rip=**<sip>**, lip=**<dip>**	IMAP Vulnerability Attack	Medium	SERVICE_EXP
	<time> <host> dovecot: imap-login: Disconnected (auth failed, one attempts in 6 secs): user=**<user>**, method=PLAIN, rip=**<sip>**, lip=**<dip>**	IMAP Password Cracking	Medium	BRUTE_FORCE
	<time> <host> dovecot: imap-login: Login: user=**<user>**, method=PLAIN, rip=**<sip>**, lip=**<dip>**, mpid=**<pid>**, TLS	IMAP Successfully Log in	High	EXFILTRATION
mongo.log	**<time>** NETWORK [id] received client metadata from **<sip>**:**<sport>** conn: application: name: "MongoDB Shell" ,driver: , os:	MongoDB interactive shell	High	CONTROL

† Due to table length limitations, we present a condensed version highlighting the essential semantic elements. The complete implementation details are available in `log_patterns.py` in our public code repository.

TABLE V: Representative examples of the classification criteria between sensitive entities and 14 attack stages within the ATT&CK framework. The complete mapping details are available in `bash_parse/utils.py` in our repository.

Attack Stage	**Sensitive Entities**
Reconnaissance	whois, nslookup, shodan, whatweb · · ·
Resource Deployment	git clone, apt install, yum install, · · ·
Initial Access	crackmapexec, medusa, bitsadmin, · · ·
Execution	bash, python, perl, ruby, · · ·
Persistence	crontab, init.d, rc.local, authorized_keys, · · ·
Privilege Escalation	sudo -l, find / -perm -4000, · · ·
Defense Evasion	service auditd stop, iptables -F, · · ·
Credential Access	hydra, cewl, hashcat, john, · · ·
Discovery	netstat, nmcli, ss, ffuf, · · ·
Lateral Movement	ssh, telnet, smbmap, smbclient, · · ·
Collection	whoami, uname, lsb_release, id, · · ·
Command and Control	-i >& /dev/tcp, -e /bin/bash, · · ·
Exfiltration	bsondump, mongodump, ftp, · · ·
Impact	systemctl, systemd, HISTFILESIZE=0, · · ·

- Event counts for inter-host episodes and intra-host behaviors during attack reconstruction.
- Number of supplementary evidence identified by MuSAR from application logs.

Fig. 14 shows the distribution of commands misclassified as benign during C-IoC identification across eight attack scenarios. File and directory manipulation commands (e.g., `cd`, `ls`, and `cat`) constitute the majority of these false negatives.

2) Attack Steps in MSAS Dataset: Ten steps in attack scenario S-1 can be described as follows:

- Reconnaissance (step-1): Initial vulnerability assessment of the portal site and proxy server using `NMAP` and `FSCAN`.
- Initial Access (step-2): Exploitation of ThinkPHP5 RCE vulnerability to compromise the web server.
- Host Enumeration (step-3): Collection of system information from the compromised web server.
- Privilege Escalation (step-4): Exploitation of Redis unauthorized access to inject SSH public key into `authorized_keys`.
- Lateral Movement (step-5): Compromise of proxy server followed by host reconnaissance.
- Persistence Establishment (step-6): Deployment of Metasploit-based reverse connection and SOCKS5 tunnel.
- Network Reconnaissance (step-7): Internal network enumeration for vulnerable services and active hosts.
- Credential Access (step-8): SSH credential compromise of employee-1 through brute-force attack.
- Lateral Movement (step-9): Access to employee-1's system and subsequent information gathering.
- Data Exfiltration (step-10): Exploitation of anonymous FTP access to extract sensitive data from employee-2's system.

Ten steps in attack scenario S-2 can be described as follows:

- Initial Access (step-1): Execution of Metasploit-generated trojan by employee-2 via phishing email through Firefox.
- Persistence Establishment (step-2): Configuration of reverse connection on port `4444` and SOCKS5 tunnel on port `1234`.
- Network Reconnaissance (step-3): Internal network enumeration via SOCKS5 tunnel with no exploitable vulnerabilities found.

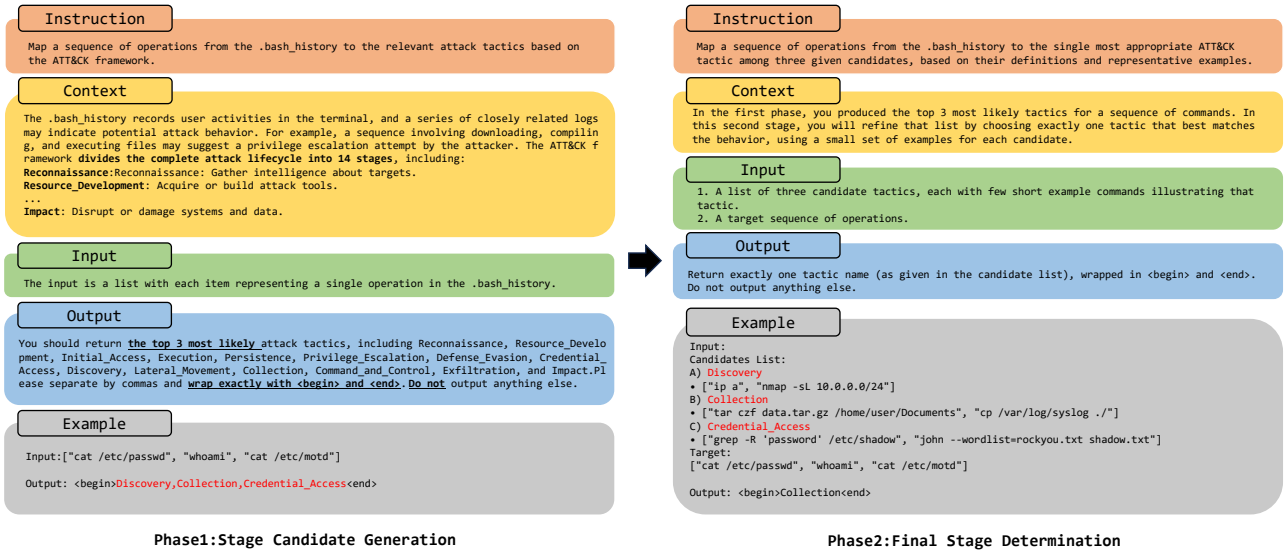

Fig. 13: Two-phase LLM-based semantic alignment for intra-host behaviors.

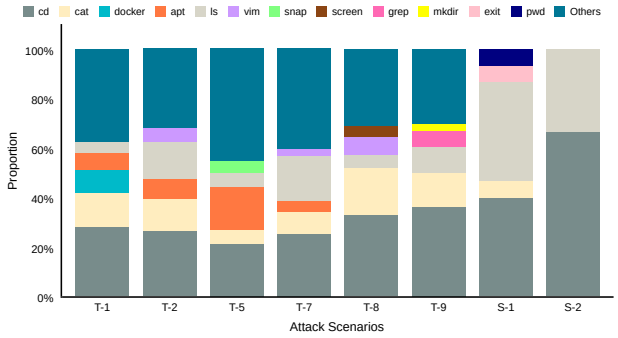

Fig. 14: Proportion of commands in the false negatives during identifying C-IoCs on eight attack scenarios.

- Host Enumeration (step-4): Collection of sensitive information from employee-2's system.
- Lateral Movement (step-5): Compromise of employee-1's system using exposed RSA keys from employee-2's `.ssh` directory.
- Network Evasion (step-6): Deployment of additional SOCKS5 tunnel on port `1235` for security bypass.
- Privilege Escalation (step-7): Authentication to OA platform using compromised employee-1 credentials.
- Web Exploitation (step-8): Compromise of web server through Struts2 OGNL injection vulnerability.
- Credential Access (step-9): Extraction of MySQL database credentials from OA configuration files.
- Data Exfiltration (step-10): Extraction of sensitive data from the database server.

3) Distribution of Attack-related Traces in the MSAS Dataset: Table VIII presents the distribution of network alarms and application logs across 10 attack steps in scenarios S-1 and S-2 on the MSAS dataset. From the table, attack-related traces account for about 44% of all lightweight security logs, with their distribution varying significantly across attack steps. Specifically, more than 75% of the traces are associated with reconnaissance attacks (e.g., step-1 in S-1 and step-8 in S-2), while lateral movement and SOCKS5 communication (e.g., step-5 in S-1 and step-5 in S-2) generate only a small number of logs, indicating higher stealth. Note that in S-2, application logs capture only two attack steps due to the attacker's extensive use of temporary shell sessions through the Metasploit framework for internal network reconnaissance and host enumeration. Under default Linux logging configurations, these shell activities are exclusively recorded for root-privileged sessions, resulting in limited visibility in application logs. Therefore, this distribution pattern in Table VIII aligns with real-world attack scenarios, demonstrating the dataset's authenticity.

C. Effectiveness of Feasibility Verification Module

To evaluate the effectiveness of the feasibility verification (FV) module, we tested MuSAR on the MSAS dataset with and without the module enabled. Table IX presents the comparative results. In scenario 1 (S-1), implementing the FV module reduced the number of attack stages requiring investigation by 40% (from 113 to 68), with only a marginal decrease in recall (from 0.9162 to 0.9106). The slight recall reduction stems from filtering the `POSSBL SCAN SHELL M-SPLOIT TCP` alarm. As we analyzed in our discussion on False Negatives and Adversarial Settings (Sec. V), this omission does not affect the final attack investigation. In Scenario 2 (S-2), the FV module reduced the number of attack graphs and attack stages by 56% (from 32 to 14) and 68% (from 667 to 214), respectively. These results demonstrate that

TABLE VI: Mapping rules for semantic alignment between 35 AIF techniques and 14 ATT&CK tactics.

ATT&CK Framework	AIF Framework
Reconnaissance	Initialize
	Target Identification
	Surfing
	Social Engineering
Resource Development	-
Initial Access	Trusted ORG. Exploit
	Public APP Exploit
	Spear Phishing
Execution	Remote Service Exploit
	Service Specific
	Arbitrary Code Execution
Persistence	Account Manipulation
Privilege Escalation	User Privilege Escalation
	Root Privilege Escalation
	Privilege Escalation
Defense Evasion	Defense Evasion
Credential Access	Brute Force Credentials
Discovery	Host Discovery
	Service Discovery
	Vulnerability Discovery
	Information Discovery
Lateral Movement	Lateral Movement
Collection	Network Sniffing
Command and Control	Command and Control
Exfiltration	Data Exfiltration
	Data Delivery
Impact	End Point DoS
	Network Dos
	Service Stop
	Resource Hijacking
	Data Destruction
	Content Wipe
	Data Encryption
	Defacement
	Data Manipulation

the FV module effectively eliminates spurious reconstructions while preserving legitimate attack graphs.

D. Performance and Limitations of LLM-based Semantic Alignment

This section provides a comprehensive evaluation of the LLM-based semantic alignment approach, examining both its performance and its inherent limitations. First, we empirically evaluate the overall alignment accuracy to establish a performance baseline. Second, we investigate the primary failure modes through a detailed case analysis. Finally, we assess the component's dependency on the underlying foundation model by comparing the performance of several prominent LLMs.

1) Performance on LLM-based semantic alignment: We evaluated the effectiveness of our LLM-based approach in interpreting and classifying intra-host security-relevant activities into corresponding attack stages. Table X presents our analysis results using both the CPTC2018 public dataset and the MSAS simulated dataset. The approach achieved an average accuracy of 98.08%, demonstrating robust performance in contextual interpretation of log sequences and precise classification of attack stages based on semantic features and entities within the logs.

2) Potential Failure Modes: Potential failure modes for LLM alignment generally fall into two main categories: semantic ambiguity and novel tools.

Semantic Ambiguity. Command interpretation often depends on contextual intent, which may not be explicitly stated. LLMs can struggle to differentiate between legitimate administrative activities and malicious operations that mirror them, as illustrated by the following command:

```
nmap -p445 -script smb-vuln-ms17-010 10.0.0.10
```

This command demonstrates semantic ambiguity, as its classification depends on the operator's intent, which ranges from legitimate vulnerability assessment to malicious probing. LLMs typically classify this command as `[Reconnaissance]` due to their training data bias, where `nmap` predominantly appears in network discovery contexts. This classification overlooks the crucial semantic significance of the `smb-vuln-ms17-010` script, which indicates potential `[Lateral Movement]` activity.

MuSAR resolves this ambiguity by correlating the command with network alarms. When the system detects the alarm `"ET EXPLOIT Possible ETERNALBLUE MS17-010 SMBv1 Exploit Attempt"`, our rule engine correctly maps the activity to `[Lateral Movement]`. This demonstrates MuSAR's effectiveness in leveraging complementary evidence sources for accurate context-aware classification.

Novel Tools. LLMs may fail to correctly classify activities involving custom binaries, obfuscated commands, or emerging attack tools absent from their training data. This limitation is demonstrated by the following command:

```
nmap -p 88 -script krb5-enum-users -script-args
userdb=/opt/userlist.txt 10.0.0.10
```

The LLM incorrectly classifies this command as `[Reconnaissance]` based on its familiarity with `nmap`, while the correct classification is `[Credential Access]`. The `krb5-enum-users` script performs username enumeration via Kerberos, which is a credential harvesting activity. This misclassification occurs because the LLM, lacking exposure to specialized security tools, defaults to classifying based on the more commonly recognized wrapper tool (`nmap`).

To address this limitation, our in-context learning approach teaches the LLM model to identify behavioral patterns rather than rely solely on tool names. We provide exemplar cases such as `john -wordlist=rockyou.txt shadow.txt` for `[Credential Access]`, which helps the model recognize characteristic patterns like dictionary

TABLE VII: Eight multi-step attack scenarios from two datasets.

Attack Scenario †	Log Count			Log Size (MB)			Categories of Attributes				Event Count		Semantic Supplement
	Network Alarms	Application Logs	Audit Logs	Network Alarms	Application Logs	Audit Logs	IPs	Services	Signatures	Durations (hours)	Episode	Behavior	
T-1	81600	27783	-	102.64	9.64	-	86	173	214	9	989	440	85
T-2	42681	75030	-	48.58	23.56	-	106	143	52	9	965	342	105
T-5	52833	20788	-	63.59	7.52	-	100	103	142	9	966	158	36
T-7	47204	32720	-	54.59	11.59	-	153	114	147	9	1422	258	112
T-8	55392	31285	-	66.59	10.33	-	86	190	155	9	899	339	106
T-9	51844	25054	-	58.59	8.28	-	129	293	154	9	1631	403	182
S-1	1881	1433	18635.9K	0.38	1.56	2905.28	40	103	146	1	62	12	8
S-2	1707	348	25253.9K	0.34	0.17	3927.61	48	77	67	1	45	1	0

† Scenarios T-1 to T-9 come from the CPTC2018 dataset, while Scenarios S-1 and S-2 are from the MSAS dataset.

TABLE VIII: Distribution of network alarms and application logs across attack steps in scenarios S-1 and S-2 on the MSAS dataset.

Attack Steps	Scenario 1		Sum.	Scenario 2		Sum.
	Network Alarms	Applica-tion Logs		Network Alarms	Applica-tion Logs	
Step-1	329	16	**345**	4	6	**10**
Step-2	36	0	**36**	7	0	**7**
Step-3	0	0	**0**	84	0	**84**
Step-4	2	0	**2**	0	0	**0**
Step-5	0	14	**14**	0	3	**3**
Step-6	438	9	**447**	409	0	**409**
Step-7	44	396	**440**	56	0	**56**
Step-8	144	9	**153**	294	0	**294**
Step-9	11	20	**31**	4	0	**4**
Step-10	8	5	**13**	12	0	**12**
Sum.	**1012**	**469**	**1481**	**870**	**9**	**879**
Percent %	**53.8%**	**32.7%**	**44.7%**	**51.0%**	**2.6%**	**42.8%**

TABLE IX: Ablation study on the feasibility verification (FV) module on the MSAS dataset.

Scenario	Method	Num. of Attack Graphs	Num. of Attack Stages	Recall
S-1	w/o FV	4	113	0.9162
	with FV	4	68	0.9106
S-2	w/o FV	32	667	0.9589
	with FV	14	214	0.9589

file arguments (`-wordlist`, `-userdb`). This pattern-based learning enables the model to prioritize behavioral indicators over tool identity, leading to better classification accuracy for specialized or novel tools.

3) Model Dependency and Performance Comparison: We evaluated MuSAR's semantic alignment performance across multiple LLMs, including two closed-source models (Qwen-turbo and GPT-4o) and two open-source models (SecGPT-14B, a cybersecurity-specific model, and its base model, Qwen2.5-14B). Our evaluation used 533 sensitive behavior sequences from the CPTC2018 dataset.

Table XI shows that the closed-source models Qwen-turbo and GPT-4o achieved superior alignment accuracies of 97.74% and 97.37%, respectively, validating the robustness of our two-stage prompt engineering approach across different commercial LLMs. The cybersecurity-specialized SecGPT-14B demonstrated competitive performance at 96.06%, significantly outperforming its base model Qwen2.5-14B (90.24%). These results indicate that domain-specific fine-tuning can substantially bridge the performance gap between open-source models and SOTA LLMs in semantic alignment tasks.

TABLE X: Performance of the LLM-based semantic alignment using Qwen-turbo.

Attack Scenarios	Sensitive Behaviors	Correctly Aligned	Accuracy
T-1	405	397	98.02%
T-2	326	318	97.55%
T-5	151	147	97.35%
T-7	243	237	97.53%
T-8	313	300	95.85%
T-9	361	355	98.34%
S-1	12	12	100.00%
S-2	1	1	100.00%
Avg.			**98.08%**

TABLE XI: Performance comparison of different LLMs for semantic alignment.

Model	Provider	Parameter Scale	Accuracy
● Qwen-turbo†	Alibaba	≈ 50B	97.74%
● GPT-4o	OpenAI	≈ 200B	97.37%
○ Qwen2.5	Alibaba	14B	90.24%
○ SecGPT*	Clouditera	14B	96.06%

● Closed-source model ○ Open-source model
† Qwen-turbo is default model for MuSAR.
* SecGPT is fine-tuned from Qwen2.5.

E. Heuristic Rules: Design Principles and Robustness

As shown in Fig. 3, MuSAR employs three distinct categories of heuristic rules:

- Anomalous pattern rules generate inter-host connections from network-based alarms. While these rules require initial configuration for deployment-specific security devices, they remain relatively stable during operation.
- Sensitive entity rules identify critical host-based events. These rules require frequent updates to address evolving attack techniques and novel command patterns.
- Stage mapping rules align the Attack Intelligence Format (AIF) with the MITRE ATT&CK framework. These expert-defined rules remain static throughout operation.

Given the dynamic nature of sensitive entity rules, we detail their lifecycle management as follows.

1) Rule Generation: The foundational rule set leverages public knowledge bases, ensuring comprehensive baseline coverage aligned with established security standards. For network-based alarms, we employ the AIF framework to map alarm signatures to attack stages. For host-based events, we derive detection rules from the Uptycs threat intelligence repository.

2) Rule Maintenance and Expansion: To maintain an evolving and scalable rule set, MuSAR employs a structured, LLM-assisted workflow that transforms the security expert's role from manual rule creation to systematic verification. This automated process initiates when security analysts detect emerging threats.

Step 1: Automated Candidate Generation. Using a standardized prompt template (i.e., "Prompt Template for Rule Generation"), analysts submit threat-related artifacts (e.g., command strings, threat report excerpts) to an LLM for automated rule generation.

Step 2: Expert Validation and Refinement. Security experts evaluate each LLM-generated rule by validating its detection accuracy, optimizing the regular expression patterns to reduce false positives, and confirming that the ATT&CK stage mapping is contextually correct.

Step 3: Integration and Testing. Following validation, MuSAR incorporates the refined rule into its rule base and evaluates it against a comprehensive benchmark dataset comprising both malicious and benign samples to ensure enhanced detection coverage while maintaining system performance metrics.

Prompt Template for Rule Generation

You are a cybersecurity expert specializing in threat detection rules. Your task is to analyze the provided information and generate a detection rule in JSON format. The rule should include a regular expression pattern, a mapping to a MITRE ATT&CK stage, and a brief description.
Input: [threat-related artifacts]
Expected output format:

```
{
    "pattern": "<regex_pattern>",
    "mitre_technique": "T####",
    "description": "<detection_logic>"
}
```

3) Robustness to Unseen Commands: Using a standardized prompt template (i.e., "Prompt Template for Classifying Unseen Commands"), MuSAR employs a dynamic classification mechanism to analyze previously unseen commands, particularly in shell command histories (`.bash_history`) where adversaries often utilize novel or obfuscated techniques. For commands that do not match existing detection rules, MuSAR leverages a real-time classification approach by querying LLMs to determine the probable MITRE ATT&CK attack stage. If the LLM cannot make a confident determination, the command will be flagged for subsequent manual review by a human analyst. This dynamic classification capability enables MuSAR to effectively respond to novel attack techniques, bridging the gap between emerging threats and rule updates while maintaining continuous detection capabilities in rapidly evolving threat landscapes.

Prompt Template for Classifying Unseen Commands

You are a cybersecurity analyst. Given the MITRE ATT&CK framework stages (Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, Impact), classify the following shell command.
If you can classify it confidently, return ONLY the most likely ATT&CK stage name. If you are uncertain, return the single word: UNCERTAIN.
Command: [Unseen Command]