Private Yet Accurate: A Decentralized Approach to System Intrusion Detection

***Abstract*—Enterprises increasingly rely on Intrusion Detection Systems (IDS) to detect malicious threats through log analysis. However, centralizing logs, which often contain sensitive data (e.g., URLs), raises serious privacy and scalability concerns. We present MIRAGE, the first privacy-preserving Provenance-based IDS (PIDS) that integrates Federated Learning (FL) with graph representation learning to match centralized detection accuracy while preserving privacy and improving scalability. Building MIRAGE is non-trivial due to challenges in federating graph-based models across clients with heterogeneous logs, inconsistent semantic encodings, and temporally misaligned data. To address these challenges, MIRAGE introduces a novel process entity categorization-based ensemble, where specialized submodels learn distinct system behaviors and avoid aggregation errors. To enable privacy-preserving semantic alignment, MIRAGE designs a dual-server harmonization framework: one server issues encryption keys, and the other aggregates encrypted embeddings without accessing sensitive tokens. To remain robust to temporal misalignment across clients, MIRAGE employs inductive GNNs that eliminate the need for synchronized timestamps. Evaluations on DARPA datasets show that MIRAGE matches the detection accuracy of state-of-the-art PIDS, reduces network communication costs by 170×, processes datasets in minutes rather than hours, and remains robust against adversarial threats.**

1. Introduction

Intrusion Detection Systems (IDS) are crucial for countering Advanced Persistent Threats (APTs) in enterprises, as highlighted by major attacks [8, 9]. To strengthen defenses, many organizations turn to Managed Security Service Providers (MSSPs). A survey [7] of over 5,000 IT professionals found that 75% of companies use MSSPs. These providers integrate with client systems and typically configure them to transmit system logs to the cloud for centralized analysis. Figure 1 illustrates this MSSP architecture.

Recently, Provenance-based IDS (PIDS) [29, 49, 50, 58, 75, 90, 100, 101, 106, 111] have proven highly effective by leveraging the rich contextual information in system logs. These systems transform system logs into provenance graphs and apply machine learning, particularly Graph Neural Networks (GNN), to learn benign behavior patterns. By continuously monitoring these patterns, PIDS can detect deviations that may indicate potential security threats. Upon detecting such anomalies, the PIDS generates alerts for further investigation.

Critical Limitations of Existing PIDS.

Despite PIDS potential, the current operational mode of MSSPs and the state-of-the-art PIDS face significant challenges in enterprise settings, which are described below.

1. PRIVACY RISKS & CENTRALIZATION. Current PIDS depend on centralized infrastructure, requiring clients to transmit logs to a central server for aggregating large datasets and enabling deep learning models to capture benign behavior. This design introduces serious privacy risks [10, 41, 95], as logs often contain sensitive information, such as URLs, IP addresses, and application usage. These concerns are supported by a recent Datadog report [4]. Moreover, training on single-machine data is insufficient. Our experiments with FLASH [90] on the DARPA OpTC dataset [3] show a 40% F-score drop when using single-host data compared to multi-host data.

2. NETWORK OVERHEADS. Many PIDS face critical challenges with network overhead and bandwidth constraints. Transmitting large volumes of logs for intrusion detection imposes high costs on users and organizations. Modern systems can produce gigabytes of logs daily [53, 55]. Our analysis of FLASH [90] and KAIROS [29] using the OpTC dataset (Section 5.3) highlights these issues. Organizations similar in scale to those in the OpTC dataset generate up to 1000 GB of logs daily, leading to significant network expenses. Users with limited bandwidth face difficulties uploading such large amounts efficiently.

3. SCALABILITY ISSUES. As the number of hosts increases, centralized PIDS suffer from log congestion, resulting in detection delays and high storage demands [34]. FLASH and KAIROS took 27.7 and 56.6 hours, respectively, to process a single day's OpTC logs. These extended processing times compromise effective threat detection and response capabilities, particularly in large-scale environments.

Combine FL with PIDS: Opportunity & Challenges.

A seemingly promising solution to mitigate privacy and scalability limitations in centralized PIDS is to integrate Federated Learning (FL) into their design. In such a federated PIDS framework, each client constructs its local provenance graph, encodes semantic attributes using local feature sets $\mathcal{F}_i$, and trains a local Graph Neural Network model GNN_i. These clients then transmit only their model updates, rather than raw logs, to a central server, where the updates are aggregated into a global model $\text{GNN}_{\text{global}}$. This approach preserves data locality and reduce bandwidth while enabling collaborative learning across multiple hosts. However, simply combining FL and PIDS introduces several new challenges that hinder effectiveness:

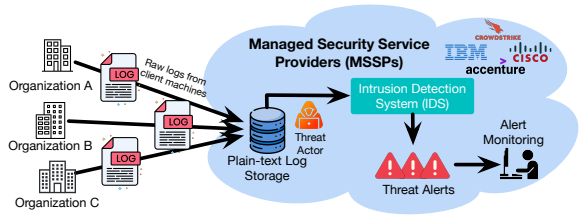


Figure 1: The MSSP architecture for intrusion detection collects plain-text system logs in a centralized storage. These system logs are then analyzed for intrusions. The architecture risks privacy leaks if a threat actor or a curious analyst within the MSSP accesses the logs.

C1 *Feature Space Heterogeneity.* Inconsistent encoding of identical features across clients leads to difficulties in model convergence and reduces the effectiveness of federated averaging. PIDS, such as FLASH [29], utilize a Word2vec model to encode semantic attributes within a provenance graph. Due to the inherent randomness of the Word2vec algorithm, identical tokens t are encoded into different vectors $v_i(t)$ by each client i, using their local feature sets $\mathcal{F}_i$. This variability disrupts the convergence and efficacy of the aggregated gobal $\text{GNN}_{\text{global}}$ model from local client models leading to suboptimal anomaly detection performance, as detailed in [115].

C2 *Data Imbalance & Heterogeneity Among Clients.* Variations in data distribution and volume across clients, often non-IID [112], pose challenges in training a global model $\text{GNN}_{\text{global}}$ that performs uniformly well. Heterogeneous data resulting from different client applications can lead to suboptimal performance [88], and data imbalances can cause models to be biased towards clients with more extensive datasets, potentially overlooking unique patterns in less-represented clients [37]. This issue is critical for PIDS, as a biased $\text{GNN}_{\text{global}}$ may result in high false alarms, undermining detection accuracy.

C3 *Temporal Misalignment.* This issue arises in PIDS which uses Temporal Graph Networks (TGN) for threat detection. Aggregating temporal graph models from clients with temporally misaligned data challenges the creation of a cohesive global model. Systems like KAIROS [29] and ORTHRUS [59] employ TGN to trace the evolution of system provenance graphs. However, federated learning's application across fragmented and misaligned client data impedes effective federated averaging, leading to improper alignment when aggregating models. This temporal constraint in TGN acts as a source of additional heterogeneity and hinders the formation of a cohesive global model $\text{GNN}_{\text{global}}$.

Our Approach and Contributions.

We present MIRAGE, a privacy-preserving PIDS that address the core challenges of applying FL to PIDS. Prior work [101] shows that graph-based models outperform traditional ML methods [31, 43] applied directly to system logs [35, 104]. Building on this insight, MIRAGE introduces **Federated Provenance Graph Learning**, combining FL's decentralized architecture with powerful graph representation learning to capture the intricate relationships among system entities. All major computations, including training, take place locally on client machines, leveraging their resources for real-time threat detection. This decentralized approach allows MIRAGE to scale effectively as more hosts are added, with each utilizing its own computing power and storage. *To the best of our knowledge,* MIRAGE *is the first system to achieve accurate, scalable, and privacy-preserving federated PIDS.*

1. PRIVACY-PRESERVING MODEL HARMONIZATION. To tackle C1, we implement a Word2vec harmonization scheme utilizing a dual-server architecture. In this setup, a central server issues encryption keys to clients, allowing them to securely encode Word2vec tokens. Subsequently, a utility server processes these encrypted tokens and corresponding vectors and to achieve a unified, privacy-preserving vector representation. This design ensures consistent semantic alignment across clients without exposing raw tokens, maintaining the privacy of sensitive process names, file paths, and network addresses.

2. ENTITY CATEGORIZATION ACROSS CLIENTS. To address C2, MIRAGE employs a novel categorization scheme for process entities, which enables consistent binning of similar activities across all clients in a privacy-preserving manner. The utility server collects encrypted process names and maps them into K_{cat} global categories using a mapping function. This standardized assignment ensures all clients use a shared category structure for training.

3. ENSEMBLE GNN LEARNING. Built on top of the categorization, we develop a GNN ensemble learning framework, where each submodel specializes in learning patterns from a specific process category. Clients align local process nodes to global bins, extract corresponding provenance subgraphs, and train category-specific models. These are aggregated across clients into K_{cat} global submodels, forming the ensemble $\text{GNN}_{\text{global}}$. This setup merges models with similar data distributions and preserves distinct local patterns.[1]

4. DECENTRALIZED ANOMALY DETECTION. To avoid C3, MIRAGE employs an inductive GNN model [47], which has been shown by prior work [90, 101, 111] to offer good performance and is not affected by temporal dependencies. Each client performs decentralized inference using the global aggregated submodels, and a node is flagged as anomalous if it is consistently misclassified across all K_{cat} submodels. This detection leverages structural patterns without requiring temporal synchronization across clients, making it inherently robust to misaligned log timestamps. It also eliminates the need for global graph construction, allowing detection to proceed efficiently and independently on each client.

5. FORMAL PRIVACY ANALYSIS. MIRAGE provides formal privacy guarantees to support its dual-server architecture. We prove that (i) model updates sent to the central

1. We compared our methods with existing solutions including FedProx [69] and FedOpt [21] for addressing heterogeneity in FL settings and found that our techniques outperform these existing solutions, as detailed in Section 5.5.

server provide indistinguishability under dataset perturbations (Central Server Privacy), (ii) the utility server cannot recover original tokens from semantic vectors (Utility Server Privacy), and (iii) malicious clients cannot infer the presence of application-specific tokens from the global model (Client-Level Privacy).

We evaluated MIRAGE using open-source datasets from DARPA, including E3 [5], E5 [1], and OpTC [15]. The evaluation covers four dimensions: (1) Accuracy, (2) Efficiency and scalability, (3) Robustness, and (4) Privacy. For accuracy, we compare MIRAGE against a vanilla privacy-preserving PIDS baseline that combines FL with a state-of-the-art (SOTA) PIDS. We find that MIRAGE significantly outperforms this baseline. Since existing SOTA PIDS already achieve near-perfect detection accuracy, *the primary goal of* MIRAGE *is to demonstrate that a privacy-preserving PIDS can be built while maintaining similar performance.* Our results show that MIRAGE achieves an average precision of 96% and recall of 97%, matching the accuracy of centralized SOTA PIDS. To evaluate efficiency and scalability, we show that MIRAGE achieves a 170-fold reduction in network communication costs compared to centralized systems. Its decentralized design enables much faster inference, bounded only by the slowest client. MIRAGE completes the full OpTC dataset in a few minutes, while existing PIDS require several hours. To evaluate robustness, we present a comprehensive analysis of MIRAGE 's resilience against adversarial attacks in Section 5.4. In Section 6, we provide a detailed analysis of the privacy guarantees offered by MIRAGE. We also present an ablation study to evaluate individual components (Appendix C), analyze the impact of differential privacy on detection accuracy (Appendix D), and compare encryption methods (Appendix E).

Availability. We will open-source our code upon publication.

2. Related work

ML-based IDS. ML techniques are widely used in threat detection. ProvDetector [100] applies Doc2Vec [66] to provenance graphs; Attack2Vec [94] uses temporally aware embeddings. DeepAid [48] classifies anomalous traffic, ProGrapher [106] combines Graph2Vec [84] and TextR-CNN [65], and StreamSpot [75] clusters graph features. Other systems [12, 13, 46, 76] explore varied embeddings; some focus on malware [26, 57, 116]. DeepLog [35] uses RNNs on logs, SIGL [50] detects software installs, Euler [62] combines GNNs and RNNs, and MAGIC [58] uses masked graph learning. MIRAGE is the first to combine federated learning with provenance-based IDS, addressing privacy, scalability, and heterogeneity.

DistDet [34] detects APTs using Hierarchical System Event Trees (HSTs) to summarize local system activity and reduce network overhead. However, this summarization comes at the cost of expressiveness and privacy. HSTs must be transmitted in plaintext to a central server, exposing sensitive execution traces without any formal privacy guarantees. More critically, HSTs encode flat, linear event sequences and lack the structural richness of provenance graphs, limiting their ability to model complex causal and multi-entity interactions. Anomalies are detected by identifying sequences absent from a reference benign HST, a simplistic approach that fails to generalize in dynamic environments and results in frequent false positives.

Rule-based PIDS. Rule-based PIDS rely on predefined rules for detecting malicious activities. Examples include Holmes [80], Rapsheet [52], Poirot [79], and CAPTAIN [99] which leverage insights from APTs to construct rule bases, achieving fewer false positives compared to ML-based systems. However, these systems face significant limitations: they cannot detect threats with novel attack signatures and require skilled security professionals to design and update the rule sets. Additionally, none of these systems ensure privacy preservation during their operation.

Federating Learning in Threat Detection. Few IDS employ federated learning, with most research centered on Network Intrusion Detection. Examples include [73], proposing FL for IoT threat detection, and [39], which introduces a differentially private system for industrial IoT. [68] presents an efficient network intrusion detection framework, while [45] addresses non-IID data issues in FL for intrusion detection. MIRAGE advances this area by being the first system to integrate FL with graph-based learning techniques for host-based threat detection using a categorization based GNN ensemble framework and secure Word2vec harmonization. XFedGraph-Hunter [96] employs FL and GNN for network intrusion detection, while FedHE-Graph [74] introduces an FL-based intrusion-detection mechanism in a single-server setting. Unlike MIRAGE, however, their approach lacks semantic harmonization which is essential for leveraging semantic feature vectors and achieving detection performance on par with centralized PIDS.

Cryptographic Techniques. Cryptographic techniques, such as Multi-Party Computation (MPC) [32] and Fully Homomorphic Encryption (FHE) [20], offer strong privacy guarantees. MPC allows multiple parties to compute a function over their inputs while keeping those inputs private, and FHE enables computations on encrypted data. However, these methods face significant challenges, including increased system complexity and scalability issues [22, 36, 40], particularly with large datasets [78]. For instance, host intrusion detection systems often process terabytes of logs, making the computational overhead of MPC or FHE impractical for real-time threat detection [67]. In contrast, MIRAGE combines a scalable encryption technique with federated learning to ensure both scalability and privacy preservation. Section 6 provides a detailed analysis of the privacy guarantees of this approach.

Privacy-preserving FL. PrivateFL [107] tackles the heterogeneity caused by differential privacy (DP) in FL systems through personalized data transformations to protect model updates from inference attacks. Similarly, [16] applies FL and DP to detect browser fingerprinting, and [33] introduces secure federated averaging using homomorphic encryption.

Poseidon [93] utilizes multiparty cryptography for privacy-preserving neural network training, while PpeFL [98] adopts local DP for FL, addressing privacy and model performance issues. Unlike these methods, MIRAGE introduces a privacy-preserving multi-model server architecture that avoids DP-induced noise, ensuring robust performance while resisting inference attacks. We provide experimental results on the issue of applying DP for the PIDS domain in Appendix D.

3. Threat Model & Assumptions

Our threat model assumes that the central server operates with integrity, conducting the federated averaging process without malicious objectives. However, we recognize the risk that the central server could compromise the privacy of client logs if raw system logs is transmitted to it [68, 73]. We also consider the possibility of a curious central server attempting membership inference attacks by utilizing the model weights.

Realism of Dual-Server (Non-Colluding) Architecture. Following precedent in cryptographic and federated learning systems [92, 103, 113], we assume the presence of a non-colluding, trusted utility server. This assumption is not only standard but also practically feasible. The utility server processes only encrypted tokens and performs no learning or aggregation itself, making its trust requirement minimal. It can be securely deployed within a Trusted Execution Environment (TEE) [77], monitored by a trusted third-party mediator [14], or hosted as a secure cloud compute instance [2] directly managed by the organization. In this architecture, the central server is responsible for aggregating and coordinating tasks, while the utility server remains under the organization's control. This separation allows the utility server to be operated by trusted internal IT teams, specialized third-party security firms, or a dedicated cloud provider, all of whom prioritize data privacy and encryption. Non-collusion could also be ensured through strict legal agreements [92].

Client-Side Threat Considerations. For individual clients, we expect that attackers could disguise their harmful activities within benign data, making it difficult to distinguish between legitimate and harmful actions. Our model also considers the threat posed by zero-day vulnerabilities. Despite these challenges, we assume that the activities of attackers will be detectable in the system's records (system logs).

Log Integrity. In line with prior studies on data provenance [29, 49, 50, 55, 58, 75, 90, 100, 101, 106, 111], our approach relies on the provenance collection system's ability to accurately record all system activities and changes. Additionally, we ensure the integrity of audit logs is maintained through the use of established tamper-resistant storage solutions, such as those described by Paccagnella et al. [86] and the Hardlog system [11]. Similar to other PIDS works, we assume the absence of any attack activities during the training phase.

TABLE 1: Key notations used in MIRAGE's design.

Notation	Definition
N	Total number of clients in federated learning.
K_{cat}	Number of categories for process entities.
$\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$	Set of all client machines.
$PGClient_i = (\mathcal{V}_i, \mathcal{E}_i)$	Provenance graph for client i, with nodes $\mathcal{V}_i$ and edges $\mathcal{E}_i$.
$GNN_{\text{global}} = \{GNN_1, \ldots, GNN_{K_{cat}}\}$	Set of global GNN models, one per category.
$w_j^{(r)}$	Weights of global GNN for category j after round r.
$\mathcal{P}_{\text{global}}$	Global set of unique process entities.
$\psi(p)$	Categorization map assigning process p to category $\mathcal{C}_j$.
y_v	True label of node v.
$\hat{y}_v^j$	Predicted label of node v by submodel j.
v_k'	Semantic embedding for attribute k, produced by a client's local Word2vec model.
$M_{\text{w2v-harm}}$	Harmonized Word2vec model that converts contextual attributes into vector space.
$\mathcal{L}^{(r)}$	Loss function value after round r.

4. Design

4.1. Overview

MIRAGE comprises six key components. Section 4.3 builds provenance graphs from system logs on each client. Section 4.4 trains local Word2vec models to encode semantic attributes. Section 4.5 harmonizes these models using encryption and a utility server. Section 4.6 categorizes process entities across clients. Section 4.7 trains category-specific GNNs locally and aggregates them via federated learning. Section 4.8 performs anomaly detection using the aggregated models. Figure 2 illustrates the architecture of MIRAGE.

4.2. Problem Statement

MIRAGE aims to detect anomalous nodes in client-specific provenance graphs $\{PGClient_1, PGClient_2, \ldots, PGClient_N\}$, which are constructed from system logs on a set of decentralized clients $\mathcal{C}$. These anomalies represent deviations from benign behavior and are indicative of potential security threats.

To achieve this, MIRAGE employs a novel adaptation of Federated Learning (FL) to collaboratively train a set of global GNN models $GNN_{\text{global}} = \{GNN_1, GNN_2, \ldots, GNN_{K_{cat}}\}$ without requiring raw log data to leave client machines. Each client locally optimizes its model by minimizing a loss function $\mathcal{L}_i(w)$ and shares only model updates Δw_i with the central server. The global model is updated iteratively through aggregation: $w = \frac{1}{N}\sum_{i=1}^{N} \Delta w_i$, where N represents the total number of clients. This decentralized approach preserves privacy while enabling the detection of threats across diverse and distributed system logs. The key notations for our system are summarized in Table 1.

4.3. Provenance Graph Constructor

MIRAGE builds a system provenance graph on each client using local system logs. It leverages built-in logging

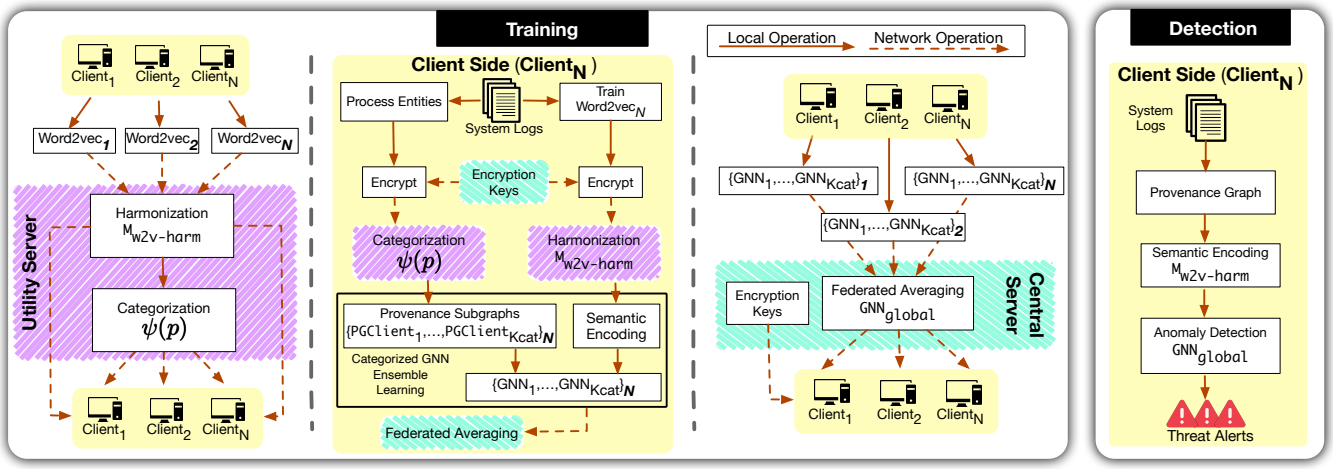


Figure 2: High-level architecture of MIRAGE. In the training phase, our system builds local provenance graphs for each client and trains an ensemble of GNN models. Prior to this, we harmonize Word2vec models in a privacy preserving manner using encryption and dual-server architecture for semantic encoding. The local GNN models participate in federated learning to develop a global GNN_{global} model, which is then utilized for anomaly detection. Notations are defined in Table 1.

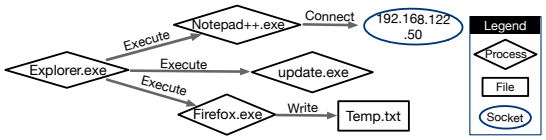


Figure 3: Data provenance graph example.

tools, such as Linux Audit [6], which record detailed system activities. These include process executions, file accesses, and network events. MIRAGE constructs a graph where nodes represent entities like processes, files, and sockets. Edges correspond to syscalls capturing interactions between them. Each node is enriched with attributes such as process names, command lines, file names, and IP addresses. Prior work [29, 90] shows these attributes help the model learn normal system behavior. Figure 3 illustrates a graph where *Explorer.exe* spawns processes like *Notepad.exe* and *Firefox.exe*, which interact with file and socket nodes.

4.4. Word2vec Model Generation

After each client machine $C_i \in \mathcal{C}$ generates its local provenance graph $PGClient_i = (\mathcal{V}_i, \mathcal{E}_i)$, the node attributes in $\mathcal{V}_i$ are converted into feature vectors for the subsequent graph learning phase. Existing systems, such as FLASH [90], have demonstrated the effectiveness of leveraging semantic attributes of nodes to enhance detection performance. System logs contain rich attributes associated with various entities, including process names, file paths, and network IP addresses. These attributes must be transformed into vector space to serve as node features for the global GNN models.

For this encoding, we employ the Word2vec model which processes different semantic attributes based on node types:

- **Process:** Process name and command-line arguments.
- **File:** File path.
- **Socket:** IP addresses and port.

For each node $v \in \mathcal{V}_i$, we extract its 1-hop subgraph by collecting all neighbors of v. We transform this node subgraph into a "document" by combining:

1) The semantic attributes of v.
2) The types of causal events captured along the edges.
3) The relevant attributes of its neighbor nodes.

These node-centric documents are then encoded into fixed-length vectors via Word2vec, which is trained on benign system logs. This approach effectively captures the semantic relationships between terms, producing dense embeddings that serve as node features for graph representation learning.

4.5. Privacy-preserving Model Harmonization

Addressing Token Variability Across Clients. Each client independently trains a Word2vec model using their local logs for feature encoding. However, before these models can be utilized to encode text attributes effectively, it is essential to contextually merge individual client Word2vec models into a unified model $M_{\text{w2v-harm}}$ for use across all clients. This unification is crucial; without it, each client would produce a different encoding, v_a^i, for identical inputs, where i denotes the client. The variability in feature vectors, $\{v_a^1, v_a^2, \ldots, v_a^N\}$, for the same attribute a across N clients, would compromise the consistency of client-based GNN models. To ensure uniformity, the feature vectors for overlapping attributes must be averaged across clients. Such averaging ensures consistency in the feature representation, enhancing the effectiveness of the federated averaging technique by maintaining uniformity in the input space for the GNN models across all clients.

Our system harmonizes only the tokens that overlap in the Word2vec models across clients. Tokens that do not overlap are not averaged and remain unchanged. While clients share common activities due to standard system-level routines, some variations and patterns are unique to each

Algorithm 1: Privacy-preserving model harmonization.

Inputs : Client Word2Vec models $\{Word2vec_1, Word2vec_2, \ldots, Word2vec_N\}$.
Output: Harmonized Word2Vec model $M_{\text{w2v-harm}}$ sent to clients.

/* Central server distribute symmetric encryption keys to each client. */
foreach *client* C_i **do**
  Send key to C_i
end
/* Clients encrypt their model tokens. */
foreach *client model* $word2vec_i$ **do**
  $Word2vec_i \leftarrow$ EncryptModelTokens($Word2vec_i$, E)
  /* Encrypt tokens using E. */
  Send $Wor2vec_i$ to Utility Server
end
/* Utility server merges encrypted models. */
$TokenVectors \leftarrow$ InitializeEmptyDictionary()
$TokenCounts \leftarrow$ InitializeEmptyDictionary()
foreach *encrypted model* $Word2vec_i$ **do**
  foreach *token* t *in* $Word2vec_i$ **do**
    $Vector \leftarrow Word2vec_i[t]$
    if $TokenVectors.HasKey(t)$ **then**
      $TokenVectors[t] \leftarrow TokenVectors[t] + Vector$
      $TokenCounts[t] \leftarrow TokenCounts[t] + 1$
    else
      $TokenVectors[t] \leftarrow Vector$
      $TokenCounts[t] \leftarrow 1$
    end
  end
end
/* Average the vectors for overlapping tokens. */
foreach *token* t *in* $TokenVectors.Keys()$ **do**
  $TokenVectors[t] \leftarrow TokenVectors[t]/TokenCounts[t]$
end
$M_{\text{w2v-harm}} \leftarrow$ NewModel($TokenVectors$, $EncryptedTokens$)
/* Constructing a new harmonized model. */
foreach *client* C_i **do**
  Send $M_{\text{w2v-harm}}$ to C_i
end
return *Harmonized model* $M_{w2v\text{-}harm}$ *has been dispatched to all clients.*

client. Non-overlapping tokens preserve these unique patterns; however, they contribute to additional heterogeneity, which cannot be eliminated through harmonization. If these unique patterns are not accurately learned, they can lead to high false alarm rates. To capture these distinct local variations between clients, we have developed a novel ensemble GNN learning framework, detailed in Section 4.7, where each submodel specializes in distinct system patterns across clients, enhancing model precision as shown in Section 5. *To better understand the degree of overlap in real-world settings, we now turn to an examination of client token overlap statistics.*

Overlap Statistics Across Clients. Our experiments with the OpTC dataset revealed that, on average, different hosts can have up to 75% overlap in process names, 41% in files, and 60% in network flows in the system logs. This finding aligns with related work [90], which shows that many system-level processes and files are common across different systems. In a centralized PIDS, these attributes are learned by a single semantic model, mapping them to the same embedding space. In contrast, a federated setting, where each client trains its own model, can introduce model bias into the token embeddings, complicating the convergence for downstream GNN models using these vectors. To address this, it is essential to harmonize overlapping tokens to provide the model with a unified understanding of the semantic information present in system logs. *However, to achieve this harmonization securely and without compromising user privacy, we employ an encryption-based strategy, as described next.*

Harmonization with Encryption. Transferring tokens, containing sensitive data like process names, file names, and IP addresses, to a central server could breach user privacy. To mitigate this, we employ a trusted utility server. Initially, the central server uses Fernet symmetric key encryption [24, 56] to generate an encryption key, which is distributed to clients. Clients then encrypt their Word2vec model tokens using the encryption key: $E(v_k) = v_k^{'}$ and send them to the utility server. This server merges the encrypted models and dispatches the unified semantic vectors back to the respective clients, who decrypt them back: $D(v_k^{'}) = v_k$. This procedure ensures that neither the central server nor the utility server can access the actual token information, assuming no collusion between the two servers. The process is detailed in Algorithm 1.

4.6. Process Entity Categorization

During our initial experiments (Table 6), we found that a single GNN_{global} cannot achieve good detection performance in an FL setting due to the diverse and heterogeneous distributions of clients data. The disparity in data distributions among N clients poses a significant challenge in effectively training a unified model that can generalize well across all clients, since the global model struggles to capture the unique characteristics and patterns inherent in each client's data. To address this limitation, we developed a comprehensive framework that organizes process entities across different clients into distinct groups. Each category is modeled by a dedicated submodel, enabling it to focus on the unique distribution and intricate patterns of its assigned category and thereby enhance overall detection performance.

The framework leverages a central utility server to orchestrate the categorization of process entities. Specifically, it proceeds in three main steps:

1) **Collection of Encrypted Process Names.** Each client $C_i \in \mathcal{C}$ transmits an encrypted list of its process names to the utility server. The server merges these into a global list of unique process entities, denoted by $\mathcal{P}_{\text{global}}$.
2) **Random Categorization.** The utility server applies the categorization map $\psi(p)$ to randomly partition all process names in $\mathcal{P}_{\text{global}}$ into K_{cat} categories. Although the partitioning is random, it is performed only once at the global level, ensuring that any process name p is consistently mapped to the same category j across all clients, i.e., $\psi(p) = j$.
3) **Distribution of Categories.** Finally, the resulting category assignments are disseminated back to each client, guaranteeing a unified and consistent mapping of processes into categorized bins across the federated network.

Figure 4 illustrates this with an example. The categorization process divides the overall learning task into sub-

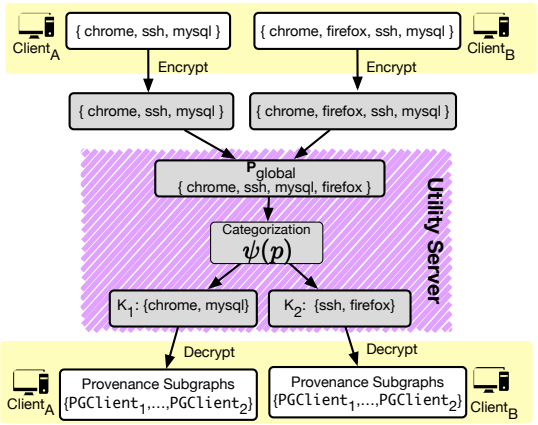


Figure 4: Secure process entity categorization workflow: Clients encrypt their process names and send them to the utility server, which categorizes them into bins and sends the results back. The clients then generate a provenance subgraph for each bin to support GNN ensemble learning. Grey boxes represent encrypted data.

tasks (submodels) by distributing them into different subsets of processes across clients in a standardized manner. As a result, the influence of clients with larger datasets is spread across multiple models. This approach reduces the risk of any single client dominating the federated learning process and promotes more balanced contributions from all participants, ultimately enhancing the system's robustness and performance.

4.7. FL with Categorized GNN Ensemble Learning

Once the categorized bins of processes have been generated, we train specialized GNN models for each bin. Each GNN_{global} submodel focuses on a standardized subset of processes across clients, identified through entity categorization. This approach effectively captures distinct patterns and structures while reducing data heterogeneity. Our method proceeds in three main steps, as detailed in Algorithm 2.

Step 1: Initialization. As in Step 1 of Algorithm 2, the central server initializes the global GNN models, GNN_{global}, with random weights $w_j^{(0)}$. The server then transmits these initial weights $w_j^{(0)}$ to all clients in $\mathcal{C}$.

Step 2: Local Model Training. Each client C_i organizes its local processes into bins based on the assigned categories, $\psi(p)$ which it gets using the methodology detailed in section 4.6. From each category bin, the client constructs a *provenance subgraph* capturing the local relationships of processes within that category. The neighborhood hop length of these subgraphs matches the number of graph convolution layers in GNN_{global}, ensuring the model has sufficient neighborhood information for each node. Each client then leverages these categorized subgraphs, along with semantic feature vectors, to train the GNN models in an unsupervised manner. Similar to the approach in FLASH, the objective of each GNN submodel is to classify every node v into its corresponding type, yielding predictions $\hat{y}_v^j$. This localized, category-focused training ensures that each submodel effectively captures the unique structures and distributions within each category for the client's data.

Algorithm 2: Training and federated aggregation of category-specific submodels.

Inputs : Global process list $\mathcal{P}_{\text{global}}$; Categorization map ψ; K_{cat}; $\mathcal{C}$;
Output: Trained global GNN models $\text{GNN}_{\text{global}}$

```
/* Step 1: Local training of submodels for each
   category.                                      */
foreach client C_i ∈ C do
    foreach p ∈ C_i do
        j ← ψ(p)
        Train GNN submodel on processes in category j from client
          C_i
    end
end
/* Step 2: Aggregation and federated averaging of
   submodels.                                     */
foreach category j ∈ {1, ..., K_cat} do
    w_j^(0) ← InitializeRandomWeights()
    foreach client C_i ∈ C do
        w_j^(i) ← ExtractWeights(C_i, j)
    end
    w_j^(r) ← FederatedAveraging({w_j^(i)})
end
return GNN_global ← { w_1^(r), w_2^(r), ..., w_Kcat^(r) }
```

Step 3: Federated Averaging and Category-Based Aggregation. As in Step 2 of Algorithm 2, the server collects the updated parameters from all N clients for each category-specific submodel. It then applies the federated averaging algorithm to merge these parameters, computing $\bar{w} = \frac{1}{N}\sum_{i=1}^{N} w_i$ for each category's global submodel. Because every submodel is trained on processes belonging to the same category across all clients, the data distributions within each submodel are more homogeneous, effectively addressing data heterogeneity. This category-based aggregation integrates knowledge gained from multiple clients, leading to robust, specialized global models. It is important to note that the central server receives only the model weights from clients; no raw system log data is transmitted to it.

The federated averaging process (Step 3 of Algorithm 2) is repeated for R rounds, concluding once there is no further reduction in the training loss $\mathcal{L}^{(r)}$. The resulting ensemble of global submodels, each attuned to a distinct process category, provides a comprehensive solution that balances the diverse data distributions across clients while preserving privacy.

4.8. GNN-based Anomaly Detection

MIRAGE employs a node level detection methodology focusing on identifying irregular nodes through the comparison of their expected and observed types. This approach is grounded in a detailed analysis of both the surrounding structures and inherent properties of the nodes, to define normal pattern baselines for various node types. Typically, entities with malicious intentions display neighborhood structures and characteristics deviating from these established norms as shown in previous work [29, 90, 106]. In op-

erational phases, the detection of anomalies that diverge from the pre-established node distribution patterns often results in their misclassification. The emergence of nodes misclassified in the system's output is indicative of potential security issues.

MIRAGE performs threat detection in a decentralized manner on clients' provenance graphs ($\{PGClient_1, PGClient_2, \ldots, PGClient_N\}$) in an organization. For a given provenance graph $PGClient_i$, MIRAGE uses the GNN_{global} $\{GNN_1, GNN_2, \ldots, GNN_{K_{cat}}\}$ trained using federated learning. Each submodel performs inference on the client's full provenance graph, utilizing the nodes' features X_v and the graph's adjacency matrix A to predict each node v's label $\hat{y}_v^j$. Let us define a misclassification indicator $\mathcal{M}(v, j)$ as

$$\mathcal{M}(v,j) = \begin{cases} 1, & \text{if } \hat{y}_v^j \neq y_v, \\ 0, & \text{otherwise.} \end{cases}$$

Accordingly, a node v is identified as an anomaly if it is misclassified by *all* submodels, i.e.,

$$\mathcal{A}(v) = \begin{cases} 1, & \text{if } \sum_{j=1}^{K_{cat}} \mathcal{M}(v,j) = K_{cat}, \\ 0, & \text{otherwise.} \end{cases}$$

This indicates that none of the submodels recognize the neighborhood structure or features displayed by v, prompting its classification as an anomaly. To regulate the frequency of alerts, we define a threshold T similar to FLASH. This parameter sets a threshold on the likelihood of a classification being considered valid, with a higher value of T implying stronger confidence and increasing the probability of identifying anomalies.

5. Evaluation

Our evaluation experiments are conducted on a machine running Ubuntu 18.04.6 LTS, equipped with a 10-core Intel CPU, NVIDIA RTX 2080 GPU, and 120 GB of memory. In our experiments, we set the federated learning rounds and the number of categorized GNN to 10 per host. Each model is trained for 20 epochs per round. We use regularization and dropout layers in our models to avoid overfitting. To evaluate MIRAGE, we address the following research questions:

- **RQ1.** How does MIRAGE compare to vanilla privacy-preserving PIDS in terms of detection performance? (Section 5.1)
- **RQ2.** How does MIRAGE compare to existing PIDS in terms of detection performance? (Section 5.2)
- **RQ3.** How scalable is MIRAGE in an enterprise-level setting with multiple host machines? (Section 5.3)
- **RQ4.** How robust is MIRAGE against adversarial attacks?
- **RQ5.** How does MIRAGE compare to existing FL solutions in addressing data heterogeneity and non-IID challenges? (Section 5.5)
- **RQ6.** What is the resource consumption of various components of MIRAGE and its end-to-end processing time on a client machine? (Appendix B)
- **RQ7.** What does the ablation study reveal about MIRAGE's effectiveness across key factors like GNN submodels, averaging rounds, federated averaging rounds, Word2vec harmonization and categorization-based GNN ensemble? (Appendix C)
- **RQ8.** How does differential privacy affect the detection accuracy of MIRAGE? (Appendix D)

Implementation. MIRAGE is developed in Python with around 6000 lines of code. It leverages the PyTorch and Torch Geometric libraries to implement the federated provenance graph learning framework. The graph learning framework uses the GraphSAGE [47] family of GNN. Our architecture consists of two graph convolution layers with a Tanh activation function in between. The last layer uses a softmax function to output class probabilities for the nodes. For implementing the Word2vec model, we employ the Gensim library. Secure communication between clients and the utility server is ensured through Python's Cryptography module. The federated averaging, semantic vector harmonization, and entity categorization modules are implemented as individual Python functions on the central and utility servers.

Datasets. We have utilized the DARPA E3 [5], E5 [1], and OpTC [3] datasets for our evaluation. These datasets contain real-world APT attacks observed in enterprise networks. The attack traces they include are stealthy, span several days, and mirror the characteristics of actual APTs. Consequently, achieving strong detection accuracy on these datasets indicates that our system can deliver comparable performance in real-world deployments. Furthermore, these datasets incorporate logs of various sizes from Linux, FreeBSD, Android, and Windows operating systems. Our system's robust detection accuracy across all datasets demonstrates its effective generalization to heterogeneous platforms with differing log sizes, reaching performance levels on par with state-of-the-art centralized PIDS. Notably, the datasets capture APT attacks of varying stealthiness, with the proportion of malicious nodes ranging from 0.05% in OpTC to 6% in E5, and E3. The low infiltration rate in OpTC underscores the system's capacity for detecting highly stealthy adversaries, whereas the more widespread attacks in E3 and E5 highlight the resilience of our approach when confronted with a higher density of malicious nodes. Collectively, these datasets serve as a strong benchmark to evaluate the scalability and adaptability of our system. Each DARPA dataset is accompanied by ground truth documents that aid in distinguishing benign events from malicious ones. For this evaluation, we employ attack labels from existing systems such as ThreaTrace, KAIROS, and FLASH. Further details regarding the datasets appear in Appendix A. ATLASv2 [91] is another recent dataset containing APT attack traces, but we did not evaluate it because it only includes data from two hosts, making it unrepresentative of a typical federated learning scenario.

Detectors for comparison. To benchmark our system, we compare against state-of-the-art PIDS. MAGIC [58] applies masked graph representation learning to identify threats. FLASH [90], another node-level system, leverages semantic

TABLE 2: Comparison of MIRAGE against vanilla privacy-preserving PIDS as baseline.

Dataset	Baseline							MIRAGE						
	Precision	Recall	F1-score	TP	FP	FN	TN	Precision	Recall	F1-score	TP	FP	FN	TN
E3-CADETS	0.64	0.99	0.78	12846	7243	4	699725	0.97	0.99	0.98	12846	389	6	706,577
E3-TRACE	0.85	0.99	0.92	67357	11390	20	2404623	0.95	0.99	0.97	67357	3196	26	2,412,811
E3-THEIA	0.69	0.99	0.81	25314	11337	38	3493999	0.95	0.99	0.97	25313	1211	49	3,504,115
OpTC	0.36	0.99	0.53	649	1142	1	1286213	0.90	0.92	0.91	596	65	54	1,287,290
E5-CADETS	0.76	0.99	0.86	40098	12356	10	1258638	0.96	0.99	0.97	40096	1844	12	1,269,150
E5-THEIA	0.73	0.99	0.84	54810	20767	270	1250910	0.99	0.99	0.99	54803	277	197	1,271,480
E5-ClearScope	0.79	0.97	0.88	13670	3491	397	79857	0.99	0.97	0.99	13679	3	388	83345

feature vectors and an embedding recycling database for enhanced detection and efficiency. As shown in Table 3, FLASH surpasses and thus serves as our primary baseline. We also include ORTHRUS [59] and KAIROS [29], which use temporal graph networks to capture system behavior over time. We exclude Streamspot [75], Unicorn [49], and ThreaTrace [101] as they are outperformed by FLASH and KAIROS. While DISDET [34], Prographer [106], and Shadewatcher [111] are notable, we exclude them as DISDET and Prographer are closed-source, and Shadewatcher relies on proprietary components, limiting reproducibility. Moreover, FLASH and ORTHRUS have already demonstrated superior performance over Prographer [106] and Shadewatcher [111]. We provide more details on why DISDET is unsuitable for comparison in Section 2. It is important to note that, similar to existing works (e.g., KAIROS, Shadewatcher, and Prographer), MIRAGE considers only three node types in provenance graphs: *processes*, *files*, and *sockets*. However, in the E3 dataset, FLASH has also been evaluated using additional node types. Therefore, we executed FLASH using these three node types to report the results in Table 3.

5.1. RQ1: Detection Performance Against a Vanilla Privacy-Preserving PIDS

We conducted experiments to analyze the detection performance of MIRAGE compared to a vanilla privacy-preserving PIDS, constructed by naively applying FL to the FLASH system. To simulate this setup, we operated FLASH in a decentralized manner: each client locally trained Word2vec to encode semantic features and then trained their own GNN models. These models were subsequently aggregated into a global model using the standard federated averaging algorithm.

We evaluated MIRAGE on the DARPA E3, E5, and OpTC datasets. E3 includes scenarios like Cadets, Theia, and Trace, each representing logs from a single host. We treated each scenario as a separate client, trained local GNN models, and performed 10 rounds of federated averaging. The global model was then evaluated on the corresponding attack logs. E5 is similarly structured but each scenario includes logs from three hosts. For the OpTC dataset, we used 10 hosts selected randomly for inclusion in our federated provenance graph learning experiment. Additionally, we conducted an enterprise-level analysis using all 1000 hosts from the OpTC dataset, the results of which are detailed in Section 5.3. In all datasets, we trained on benign logs and evaluated on attack logs that appear chronologically after. For example, OpTC contains six days of benign and three days of attack logs, and we evaluated MIRAGE across all attack days. Detection metrics matched those used by prior node-level detectors, including ThreaTrace, MAGIC, and FLASH.

The results are shown in Table 2. MIRAGE consistently outperforms the vanilla FL FLASH across all detection metrics. This performance gain is due to MIRAGE 's use of Word2vec harmonization and categorization-based ensemble learning, which effectively handle data heterogeneity. In contrast, the vanilla FL FLASH lacks mechanisms to address such heterogeneity, resulting in degraded detection performance. We observed similar degradation with other centralized PIDS as well, which can be attributed to the absence of specialized mechanisms for handling heterogeneity in decentralized model training settings.

5.2. RQ2: Detection Performance Against SOTA PIDS

We conducted experiments to assess how MIRAGE compares with other systems in terms of detection performance. We used the same experimental setup and datasets described in **RQ1**. Table 3 reveals that MIRAGE's performance on these datasets is comparable to that of MAGIC, FLASH, KAIROS, and ORTHRUS despite the data heterogeneity, diverse log patterns, and data imbalance contained within each E3, E5, and OpTC datasets. This underscores MIRAGE's capability to maintain robust detection performance amidst such heterogeneity. KAIROS's evaluation, based on a coarser time-window granularity compared to the node-level granularity of FLASH and MIRAGE, poses a challenge for direct comparison. Nevertheless, our results remain competitive with KAIROS. In some scenarios, MIRAGE exhibits slightly lower precision than centralized PIDS systems. This is primarily due to the decentralized nature of training, where each client learns from a limited and locally biased view of system behavior. Without access to the full global context, certain rare benign patterns may appear anomalous to local

TABLE 3: Comparison of MIRAGE with SOTA PIDS. Prec.: Precision; Rec.: Recall; F1.: F1-Score. While FLASH performs slightly better, MIRAGE offers strong privacy and scalability through decentralization. Refer to SOTA PIDS papers for their FP/FN details. The fraction in parentheses for Mirage indicates how many systems (out of the total compared) it outperforms or matches on that metric.

Dataset	MAGIC [58]			FLASH [90]			KAIROS [29]			ORTHRUS [59]			MIRAGE		
	Prec.	Rec.	F1.	Prec.	Rec.	F1.	Prec.	Rec.	F1.	Prec.	Rec.	F1.	Prec.	Rec.	F1.
E3-CADETS	0.94	0.99	0.97	0.99	0.99	0.99	0.80	1.00	0.89	0.52	0.37	0.43	0.97 (**3/4**)	0.99 (**3/4**)	0.98 (**3/4**)
E3-TRACE	0.99	0.99	0.99	0.99	0.99	0.99	–	–	–	–	–	–	0.95 (**0/2**)	0.99 (**2/2**)	0.97 (**0/2**)
E3-THEIA	0.98	0.99	0.99	0.99	0.99	0.99	0.91	1.00	0.95	0.81	0.40	0.54	0.95 (**2/4**)	0.99 (**3/4**)	0.97 (**2/4**)
OpTC	–	–	–	0.93	0.92	0.93	0.84	1.00	0.91	–	–	–	0.90 (**1/2**)	0.92 (**1/2**)	0.91 (**1/2**)
E5-CADETS	0.00	1.00	0.00	0.97	0.99	0.98	1.00	1.00	1.00	0.17	0.02	0.03	0.96 (**2/4**)	0.99 (**3/4**)	0.97 (**2/4**)
E5-THEIA	0.00	0.00	0.00	0.99	0.99	0.99	0.67	1.00	0.80	0.87	0.19	0.31	0.99 (**4/4**)	0.99 (**3/4**)	0.99 (**4/4**)
E5-ClearScope	0.00	1.00	0.00	0.99	0.96	0.98	0.67	1.00	0.80	0.33	0.08	0.13	0.99 (**4/4**)	0.97 (**3/4**)	0.99 (**4/4**)

models, leading to a modest increase in false positives.[2] Beyond detection performance, we also highlight MIRAGE's qualitative advantages, including its privacy-preserving features and decentralized, scalable operation. These aspects underscore the value of MIRAGE in contrast to centralized systems, emphasizing its high practicality in real-world deployments.

5.3. RQ3: Scalability in Enterprise settings

We evaluated MIRAGE 's scalability using the OpTC dataset, which simulates a large enterprise environment with numerous hosts. We compared MIRAGE to centralized systems like FLASH and KAIROS, focusing on network and processing overhead. Additional discussion on scaling FL to large networks appears in Appendix F.

Network Overhead:. We estimate this cost for the FLASH and KAIROS system using the OpTC dataset. Each host within OpTC produces approximately 1GB of audit logs daily, equating to nearly one million audit events. For an organization with 1,000 hosts, the total daily log volume would be 1,000 GB. This data volume necessitates transmission over the network to a central server operating the PIDS. In contrast, MIRAGE achieves a significant reduction in these overheads. The only network expenses for MIRAGE arise from the transmission of the GNN and Word2vec models; the GNN model is roughly 13KB, while the average Word2vec model is 6 MB. Thus, the communication cost with the central server would be 12.70 MB, and for the utility server, it would be 5.86 GB. Hence, the network latency of model communications in MIRAGE is minimal compared to centralized systems where the raw system logs need to be sent over the network. Ultimately, MIRAGE results in a 170-fold decrease in communication costs compared to centralized PIDS.

Processing Overhead:. Additionally, FLASH processes one million events in about 100 seconds, implying that processing events from 1,000 clients would necessitate approximately 27.7 hours. In contrast, KAIROS processes 57,000 events in 11.6 seconds, leading to a processing time of 204 seconds for one million events. Consequently, processing data from 1,000 clients with KAIROS would require around 56.6 hours. Compared to existing systems, MIRAGE processes client logs in a decentralized manner and the total processing time is bounded by the client with the most log data; it will only take approximately 3 minutes to run inference on the complete OpTC dataset. The central and utility servers conduct a simple mean operation on the models, taking only a few seconds.

5.4. RQ4: Adversarial Attacks Analysis

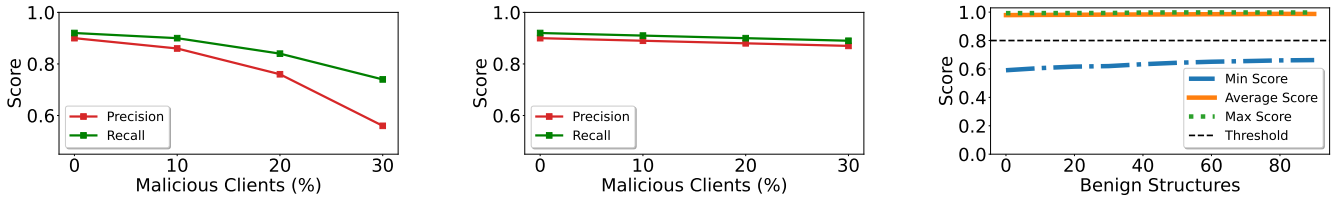
In this RQ, we evaluate MIRAGE's robustness against mimicry, model poisoning, gradient-based, and bin-level inference attacks. Membership inference is covered in Section 6.

Mimicry Attacks. We evaluated MIRAGE 's resilience to the mimicry attack proposed by Goyal et al.[44], which embeds benign substructures into malicious graphs to evade detection. Using the E3 dataset, similar to FLASH for fair comparison, our findings (shown in Figure 5c) reveal that our detector remains robust; the integration of benign structures has a minimal impact on anomaly scores. Our system's superior robustness is due to our process-categorization-based ensemble GNN architecture, which allows model specialization for different system entities, enhancing detection of structural changes. In contrast, FLASH uses a single generalized model, making it more prone to missing subtle changes. As demonstrated by Rehman et al., FLASH exhibits an initial drop in anomaly scores under attack, allowing evasive nodes to slip through, a vulnerability not observed in MIRAGE.[3]

Model Poisoning Attacks. These occur when one or more malicious clients submit corrupted local model updates to

2. We excluded KAIROS on E3 Trace and ORTHRUS from both E3 Trace and OpTC benchmarks due to missing results in their original publications. Further, unlike FLASH, both KAIROS and ORTHRUS are built on Temporal Graph Network (TGN) architectures, which are highly sensitive to time-window selection and require extensive hyper-parameter tuning to achieve stable performance. Despite multiple efforts, we were unable to reproduce satisfactory results on these datasets using their open-source implementations.

3. PROVNINJA [81] is another mimicry attack that uses benign process profiles to find adversarial evasion strategies. It faces challenges against PIDS like FLASH and MIRAGE due to their multimodal architecture, which combines Word2vec for featurization and GNN for anomaly detection. This complexity makes it difficult for attackers to create effective evasion strategies without accessing Word2vec directly or excessively querying the model, which conflicts with the black box nature of the assumed threat model of PROVNINJA.



(a) Model poisoning: FedAvg aggregation. **(b)** Model poisoning: Multi-Krum aggregation. **(c)** Adversarial mimicry attack

Figure 5: Adversarial attacks analysis.

the central server during training, thereby degrading the model's performance. Existing methods typically assume an attack-free training phase, providing no defense against such attacks. We conducted experiments on the OpTC dataset to assess the impact of poisoning attacks on our system, and we also evaluated how robust aggregation methods, such as Multi-Krum [82], can enhance resilience.

Multi-Krum compares each client's gradient with those of other clients, retaining only the most consistent updates. Since malicious updates must deviate significantly from benign ones to degrade performance, Multi-Krum effectively identifies and discards them as outliers. Figure 5 shows our experimental results using both FedAvg and Multi-Krum. With simple federated averaging, malicious noise critically affects the model by disrupting the benign distribution it learns. In contrast, Multi-Krum isolates and removes erroneous updates from malicious clients, preserving the global model. Notably, Multi-Krum assumes that fewer than one-third of all clients are malicious; therefore, in our experiments, we tested with a maximum of 30% compromised clients.

Gradient-based Attacks. These attacks [27] exploit detailed knowledge of the target model, including its architecture and parameters, to calculate and apply malicious perturbations. Such white-box access allows an attacker to compute precise gradients that indicate how inputs should be modified to degrade the model's performance. Other attacks tend to be black-box in nature, relying on iterative, query-based techniques to influence the model's decisions. However, these repeated computations and queries run counter to the attacker's aim of remaining inconspicuous, as they generate substantial activity and leave a significant footprint. Consequently, such attacks are often impractical in real-world scenarios. Several defenses can be employed during model training to bolster the system's resilience against these threats. Adversarial training [97] is one effective strategy in which the model is trained on perturbed input data, thereby increasing its robustness against these attacks.

Bin-level Inference Attacks. In addition to the privacy risks outlined in Section 6, we identify another potential avenue for information leakage: bin-level inference attacks during training. In this scenario, a malicious client attempts to infer information about system entities held by other clients. It does so by manipulating the tokens it submits and observing changes in the utility server's harmonized vectors. Since overlapping token vectors are averaged, any noticeable change in the returned vector indicates that the token is shared across clients within the organization. However, unlike the central and utility server inference risks, this attack does not compromise client identities and only the presence of a token in the system is revealed, preserving anonymity. Furthermore, our threat model 3, consistent with prior PIDS works, assumes a benign training environment free from adversarial clients. Bin-level inference attacks are thus only feasible during the training-time harmonization phase and not during deployment. We leave the development of federated learning methods robust to such adversarial training settings as an important direction for future work.

5.5. RQ5: Comparison with existing FL solutions

We conducted experiments to evaluate the efficacy of existing federated learning solutions that address data heterogeneity and the non-IID problem. Specifically, we examined FedProx [69] and FedOpt [21]. FedProx mitigates client heterogeneity by adding a proximal term to the local loss function, penalizing deviations from the global model and thereby reducing the impact of statistical heterogeneity. In contrast, FedOpt uses a server-side optimizer to aggregate updates from distributed clients, enhancing convergence. We compared these techniques with standard FedAvg and the MIRAGE variant of FedAvg, which incorporates semantic harmonization and categorization-based GNN ensemble learning. We used the OpTC, E3, and E5 datasets to conduct this experiment. Table 4 presents the evaluation results. We averaged the results for E3 and E5 across individual scenarios (Cadets, Theia, ClearScope) due to space constraints. Both FedProx and FedOpt outperform FedAvg; however, neither matches the performance of MIRAGE.

FedProx's penalty on global and local deviations focuses the model on patterns common across all clients, preventing it from learning client-specific patterns. Consequently, using this global model for anomaly detection on individual clients often yields numerous false alarms and poor detection performance. Moreover, while FedOpt can effectively aggregate model updates using an optimizer rather than simple averaging, updates that are highly disparate, a common scenario in system logs due to diverse client activities, cause FedOpt to fail in harmonizing the updates. As a result, it loses information on important benign patterns, leading to low precision and recall.

TABLE 4: Federated averaging algorithms comparison.

Methods	OpTC			E3			E5		
	Prec.	Rec.	F-score	Prec.	Rec.	F-score	Prec.	Rec.	F-score
FedAvg	0.36	0.99	0.53	0.73	0.99	0.84	0.76	0.98	0.86
FedProx	0.48	0.95	0.64	0.79	0.98	0.87	0.82	0.95	0.88
FedOpt	0.52	0.90	0.74	0.81	0.97	0.88	0.84	0.92	0.87
MIRAGE	0.90	0.92	0.91	0.96	0.99	0.97	0.98	0.98	0.98

In contrast, our system employs an ensemble learning framework. First, it categorizes system patterns into standardized bins via process entity categorization. Next, GNN submodels learn the data distribution within each bin. Submodels with similar learned distributions are then averaged, preventing the conflation of unique client patterns and improving precision. Moreover, our semantic vector harmonization reduces heterogeneity in the GNN feature vectors, making the updates less disparate and improving model convergence in comparison to other techniques.

6. Privacy Preservation Analysis

Within MIRAGE, potential privacy compromises arise if either the central server, utility server or a malicious client can infer specific details about individual clients' logs, such as the applications in use or particular attributes like filenames and IP addresses. For each of the three components we present the proofs with strong privacy guarantees below:

Theorem 1 (Central Server Privacy)**.** *For any two provenance graphs $PGClient_1$, $PGClient_2$ that differ only in the inclusion of a single node x and its ℓ-hop attributed neighborhood (i.e., the local subgraph centered at x, including all nodes in $\mathcal{N}_\ell(x)$, edges, and associated attributes), the probability that an adversary $\mathcal{A}$, observing only the GNN model updates received by the central server, can determine which graph was used deviates from random guessing by at most a negligible function $\epsilon(n)$, where n is the security parameter representing the randomness complexity of the client-side training process.*

Proof. Let $x \in \mathcal{V}_i$ denote a node in the client provenance graph $PGClient_i = (\mathcal{V}_i, \mathcal{E}_i)$. Define its ℓ-hop attributed neighborhood as the induced subgraph that includes: - the node x, - all nodes within ℓ hops (i.e., $\mathcal{N}_\ell(x)$), - all edges between them, and - their associated node and edge attributes.

Let $PGClient_1$ and $PGClient_2$ be two graphs that are identical except that $PGClient_1$ includes node x and its ℓ-hop attributed neighborhood, while $PGClient_2$ does not.

Each client transforms the contextual attributes using the harmonized embedding model $M_{\text{w2v-harm}}$, which produces vector representations for both nodes and edges. These embeddings are input to an ensemble of category-specific GNNs $\{GNN_1, \ldots, GNN_{K_{\text{cat}}}\}$, each trained independently on its category. The resulting local gradient updates are sent to the central server, where they are aggregated into global model weights $w_j^{(r)}$ for each category j.

Let $GNN_j(PGClient_i; r)$ denote the client-side training of submodel j on provenance graph $PGClient_i$ using randomness seed r, which controls embedding initialization, mini-batch sampling, and optimizer dynamics. Even for the same graph, different runs may yield different model updates due to this local randomness.

Let P_1 and P_2 denote the distributions over model updates resulting from training on $PGClient_1$ and $PGClient_2$, respectively. Their statistical distance is defined as:

$$\Delta(P_1, P_2) = \sup_S \Big| \Pr\left[GNN_j(PGClient_1) \in S\right] - \Pr\left[GNN_j(PGClient_2) \in S\right] \Big|,$$

where the supremum is taken over all measurable subsets S of the update space.

Even though the only change between the graphs is node x, the message-passing nature of GNNs causes its ℓ-hop neighbors to also influence and be influenced by the presence of x. Since embeddings propagate through the graph, the resulting update signal is not isolated to x, but distributed across the receptive field. This non-locality, coupled with random initialization and category-specific GNNs, makes the change statistically hard to detect.

Additionally, the number of possible attributed neighborhoods grows exponentially. Given node attribute vocabulary $\mathcal{A}_v$ and edge attribute vocabulary $\mathcal{A}_e$, the number of candidate neighborhoods centered at x satisfies:

$$|\mathcal{S}(x)| \in \mathcal{O}\left(|\mathcal{A}_v|^{|\mathcal{N}_\ell(x)|} \cdot |\mathcal{A}_e|^{|\mathcal{N}_\ell(x)|}\right).$$

Thus, an adversary attempting to infer the presence of x would need to simulate an exponential number of structurally and semantically plausible neighborhoods, without access to the embedding function $M_{\text{w2v-harm}}$ or the client's randomness seed.

Consequently, the adversary's distinguishing advantage is bounded by:

$$\left|\Pr[\mathcal{A}(w_j^{(r)} \text{ from } PGClient_1) = 1] - \tfrac{1}{2}\right| \leq \epsilon(n),$$

where $\epsilon(n)$ is negligible in the security parameter n. That is, for every polynomial $p(n)$, there exists n_0 such that for all $n > n_0$, $\epsilon(n) < \frac{1}{p(n)}$.

Therefore, MIRAGE ensures indistinguishability-based privacy under server-side observation of GNN model updates, even when client graphs differ by a single node and its attributed neighborhood. □

Theorem 2 (Utility Server Privacy)**.** *Let $M_{w2v\text{-}harm} : \mathcal{P}_{global} \to \mathbb{R}^d$ denote the client-side embedding function that maps input tokens $p \in \mathcal{P}_{global}$ to high-dimensional vectors $e \in \mathbb{R}^d$. In* MIRAGE*, each client independently applies $M_{w2v\text{-}harm}$ to locally generate semantic embeddings and encrypts all tokens before transmission. The utility server receives only encrypted tokens and their corresponding embeddings. Then, for any embedding e observed by the utility server, the probability that an adversary $\mathcal{A}$ can correctly identify the original token p such that $e = M_{w2v\text{-}harm}(p)$ is at most $\frac{1}{|\mathcal{P}_{global}|} + \epsilon(d)$, where $\epsilon(d)$ is negligible in the embedding dimension d.*

Proof. Each client applies the local embedding function $M_{\text{w2v-harm}}$ to convert input tokens $p \in \mathcal{P}_{\text{global}}$ into semantic vectors $e = M_{\text{w2v-harm}}(p) \in \mathbb{R}^d$, and encrypts the original tokens before sending them to the utility server. Thus, the utility server receives only encrypted identifiers and their associated semantic embeddings, with no access to the plaintext tokens or the internals of $M_{\text{w2v-harm}}$.

The embedding function $M_{\text{w2v-harm}}$, trained using a distributional objective (e.g., skip-gram with negative sampling), is inherently non-injective. Its goal is to preserve contextual similarity, not one-to-one invertibility. Formally, there exist distinct tokens $p_1, p_2 \in \mathcal{P}_{\text{global}}$, with $p_1 \neq p_2$, such that $M_{\text{w2v-harm}}(p_1) \approx M_{\text{w2v-harm}}(p_2)$. Consequently, the preimage set of a given embedding e is:

$$M^{-1}_{\text{w2v-harm}}(e) = \{p \in \mathcal{P}_{\text{global}} \mid M_{\text{w2v-harm}}(p) \approx e\},$$

which may contain multiple semantically related tokens. This ambiguity arises because $M_{\text{w2v-harm}}$ typically involves non-linear transformations and dimensionality reduction.

Let $\text{Rev}(M_{\text{w2v-harm}}, e)$ denote a hypothetical reverse function attempting to infer p from e. Since the utility server has no access to the original training data or encryption keys, its best strategy is effectively bounded by random guessing:

$$\Pr[\text{Rev}(M_{\text{w2v-harm}}, e) = p] \leq \frac{1}{|\mathcal{P}_{\text{global}}|} + \epsilon(d),$$

where $\epsilon(d)$ becomes negligible as d increases. That is, for every polynomial $p(d)$, there exists d_0 such that for all $d > d_0$, $\epsilon(d) < \frac{1}{p(d)}$.

Therefore, MIRAGE guarantees that the probability of reverse-engineering semantic embeddings at the utility server is negligible, ensuring strong privacy protection. □

Theorem 3 (Client-Level Privacy)**.** *In* MIRAGE*, let $w_j^{(r)}$ denote the global model weights for category j after round r, obtained via aggregation of updates from clients $\mathcal{C} = \{C_1, \ldots, C_N\}$. Suppose a malicious client $C_{adv} \in \mathcal{C}$ attempts to determine whether a specific process entity $p \in \mathcal{P}_{global}$ was used by another client. Then, under* MIRAGE *'s federated aggregation design, C_{adv} cannot attribute the presence of p to any specific client, and client-level privacy is preserved.*

Proof. MIRAGE uses standard federated averaging to compute global model weights $w_j^{(r)}$ by aggregating local model updates from all clients in $\mathcal{C}$. These local updates are never shared directly among clients; only their aggregated result is transmitted to participating parties.

As a result, a malicious client C_{adv} receives only the final aggregated model and has no visibility into individual contributions. The influence of any specific process entity $p \in \mathcal{P}_{\text{global}}$ on $w_j^{(r)}$ is diluted across all participating updates. Without access to isolated gradients or update traces, C_{adv} cannot determine whether p originated from any specific client.

This anonymity-by-aggregation ensures that model contributions cannot be disaggregated or linked to specific sources. Therefore, MIRAGE preserves client-level privacy against attribution and token inference attacks under the federated aggregation setting. □

Therefore, the architecture of MIRAGE robustly defends against inference attacks from both servers and malicious clients. Its privacy guarantees rely on decentralized learning, semantic embedding ambiguity, and the assumption of non-colluding servers, as upheld by prior works [92, 103] and established in our threat model 3.

7. Discussion & Future Work

Combining FL with GNN. APTs involve causally linked steps across system entities, best modeled by provenance graphs [55]. Traditional log-level systems [35, 70, 104] fail to capture these relationships, whereas graph learning techniques effectively detect such patterns [29, 58, 90]. By integrating FL with GNN, MIRAGE enables decentralized, privacy-preserving intrusion detection without sacrificing accuracy.

Unseen Tokens in Semantic Encoder. MIRAGE uses Word2vec models to encode semantic attributes, but these models only generate embeddings for previously seen tokens, degrading feature quality for new nodes with unseen tokens. Node representations in MIRAGE combine semantic attributes with neighboring system calls, but for unseen tokens, system calls dominate, as in ThreaTrace [101]. More frequent Word2vec retraining or adopting subword-level models like fastText [61], which build embeddings from subword units, can improve coverage. Tokenization methods like Byte-Pair Encoding [18] further aid by breaking unknown words into smaller components.

Scalability Bottlenecks. As MIRAGE scales to large enterprise networks, two key bottlenecks emerge: (i) the utility server may face throughput limits due to processing encrypted tokens from all clients, and (ii) aggregating frequent model updates from many clients risks state explosion, increasing bandwidth and computational costs. MIRAGE mitigates these via lightweight models, ensemble learning over categorized system activities, and submodel aggregation. To further scale, we plan to explore hierarchical federation [114], model compression [30], selective update aggregation [108], and robust communication protocols [64].

GraphSage Over Others. We adopt GraphSAGE as our base GNN due to its inductive capability and efficient minibatch training, both well-suited for federated settings. Compared to GCN, which requires full-graph access, and TGN, which imposes a temporal constraint that complicates federated learning, GraphSAGE offers a better balance of scalability, flexibility, and performance.

Alert Investigation. Validating alerts in systems like MIRAGE is critical for reliability and avoiding alert fatigue [51]. Traditionally, security analysts manually review alerts based on activities within local provenance graphs, but this process is time-consuming, error-prone, and lacks scalability and privacy. Privacy-preserving techniques such

as Secure Multi-party Computation (SMC) [42], Homomorphic Encryption (HE) [109], and Zero-Knowledge Proofs (ZKP) [38] can address these challenges. SMC enables collaborative alert validation while preserving input privacy, HE ensures confidentiality during encrypted data analysis, and ZKP allows one party to prove an alert's validity without revealing additional information. We identify privacy-preserving alert verification as a promising research direction. We leave it to future work to develop methods for privately sharing alert data with a central server, enabling analysts to perform attack analysis.

Explainability. Deep learning models in PIDS, including MIRAGE, act as black boxes, making their decisions hard to interpret. Like other PIDS [29, 90, 106], MIRAGE struggles with this, limiting adoption versus rule-based systems. Existing explainability methods [17, 19, 25, 54] mainly assess feature importance. But modern PIDS use complex pipelines e.g., FLASH uses Word2vec for textual features and GNN for anomaly detection reducing existing methods' effectiveness. Explainable PIDS remain a key research direction, with recent LLM advances [28] offering promising avenues.

Log Retention. Effective retention [52, 102] is essential for investigating long-running APTs. In decentralized systems like MIRAGE and DISDET, local storage constraints limit log history. To mitigate this, MIRAGE can integrate secure, tamper-resistant cloud storage [11, 63] for backups. During investigations, logs linked to alerts can be retrieved and sanitized [87] to preserve privacy while enabling detailed analysis.

Concept Drift. Concept drift occurs when the data distribution of the system evolves over time, potentially invalidating the patterns learned by MIRAGE during training. Emerging system activities can result in new benign behaviors being misclassified as anomalies. To address this, periodic retraining with recent data is essential to update the models. Strategies from recent works [23, 60, 71] provide effective approaches for mitigating concept drift.

8. Conclusion

We introduced MIRAGE, a privacy-preserving intrusion detection system that combines federated provenance graph learning with secure Word2vec harmonization and process entity categorization-based GNN ensemble training. MIRAGE addresses data heterogeneity, preserves client privacy, and reduces network overhead. Evaluations on large datasets show that MIRAGE matches centralized PIDS in accuracy, scales better, and remains robust against adversarial attacks.

References

[1] DARPA Engagement 5. https://github.com/darpa-i2o/Transparent-Computing/tree/master.

[2] What is an Instance in Cloud Computing? https://scaleyourapp.com/what-is-an-instance-in-cloud-computing-a-thorough-guide/.

[3] DARPA OPTC. https://github.com/FiveDirections/OpTC-data.

[4] Audit Logging Overview. https://www.datadoghq.com/knowledge-center/audit-logging/.

[5] DARPA Engagement 3. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.

[6] The Linux audit daemon. https://linux.die.net/man/8/auditd.

[7] IT security economics. https://www.kaspersky.com/blog/it-security-economics-2020-part-4/.

[8] NotPetya Attack. https://en.wikipedia.org/wiki/Petya_and_NotPetya.

[9] The SolarWinds Cyber-Attack: What You Need to Know. https://www.cisecurity.org/solarwinds/.

[10] R. Aghili, H. Li, and F. Khomh. An empirical study of sensitive information in logs, 2024. URL https://arxiv.org/abs/2409.11313.

[11] A. Ahmad, S. Lee, and M. Peinado. Hardlog: Practical tamper-proof system auditing using a novel audit device. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2022.

[12] S. Aljawarneh, M. Aldwairi, and M. B. Yassein. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, 25, 2018.

[13] 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 *USENIX Security Symposium*, 2021.

[14] J. Alwen, J. Katz, Y. Lindell, G. Persiano, A. Shelat, and I. Visconti. Collusion-free multiparty computation in the mediated model. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Springer, 2009.

[15] M. M. Anjum, S. Iqbal, and B. Hamelin. Analyzing the usefulness of the darpa optc dataset in cyber threat detection research. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 2021.

[16] M. S. M. S. Annamalai, I. Bilogrevic, and E. De Cristofaro. Fp-fed: Privacy-preserving federated detection of browser fingerprinting. *arXiv preprint arXiv:2311.16940*, 2023.

[17] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert systems with applications*, 186, 2021.

[18] A. Araabi, C. Monz, and V. Niculae. How effective is byte pair encoding for out-of-vocabulary words in neural machine translation? *arXiv preprint arXiv:2208.05225*, 2022.

[19] C. Ardito, Y. Deldjoo, E. Di Sciascio, and F. Nazary. Revisiting security threat on smart grids: Accurate and interpretable fault location prediction and type classification. In *ITASEC*, 2021.

[20] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand. A guide to fully homomorphic encryption. *Cryptology ePrint Archive*, 2015.

[21] M. Asad, A. Moustafa, and T. Ito. Fedopt: Towards communication efficiency and privacy preservation in federated learning. *Applied Sciences*, 10(8), 2020.

[22] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013.

[23] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

[24] M. U. Bokhari and Q. M. Shallal. A review on symmetric key encryption techniques in cryptography. *International journal of computer applications*, 147(10), 2016.

[25] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the first workshop on machine learning for computing systems*, 2018.

[26] S. S. Chakkaravarthy, D. Sangeetha, and V. Vaidehi. A survey on malware analysis and mitigation techniques. *Computer Science Review*, 32, 2019.

[27] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1), 2021.

[28] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3), 2024.

[29] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han. Kairos: Practical intrusion detection and investigation using whole-system provenance. In *IEEE Symposium on Security and Privacy (SP)*, 2024.

[30] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53, 2020.

[31] K. Chowdhary and K. Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, 2020.

[32] R. Cramer, I. B. Damgård, et al. *Secure multiparty computation*. Cambridge University Press, 2015.

[33] V. A. Dasu, S. Sarkar, and K. Mandal. Prov-fl: Privacy-preserving round optimal verifiable federated learning. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, 2022.

[34] 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.

[35] M. Du, F. Li, G. Zheng, and V. Srikumar. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[36] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, 2001.

[37] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1), 2020.

[38] U. Fiege, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 1987.

[39] O. Friha, M. A. Ferrag, M. Benbouzid, T. Berghout, B. Kantarci, and K.-K. R. Choo. 2df-ids: Decentralized and differentially private federated learning-based intrusion detection system for industrial iot. *Computers & Security*, 127, 2023.

[40] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009.

[41] S. Ghiasvand and F. M. Ciorba. Anonymization of system logs for preserving privacy and reducing storage. In *Advances in Information and Communication Networks: Proceedings of the 2018 Future of Information and Communication Conference (FICC), Vol. 2*. Springer, 2019.

[42] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110), 1998.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11), 2020.

[44] 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.

[45] W. Guo, Z. Yao, Y. Liu, L. Zhang, L. Li, T. Li, and B. Wu. A new federated learning model for host intrusion detection system under non-iid data. In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2023.

[46] M. Gyanchandani, J. Rana, and R. Yadav. Taxonomy of anomaly based intrusion detection system: a review. *International Journal of Scientific and Research Publications*, 2, 2012.

[47] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[48] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin. Deepaid: Interpreting and improving deep learning-based anomaly detection in security applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[49] X. Han, T. Pasqueir, A. Bates, J. Mickens, and M. Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *Network and Distributed System Security (NDSS)*, 2020.

[50] X. Han, X. Yu, T. Pasquier, D. Li, J. Rhee, J. Mickens, M. Seltzer, and H. Chen. Sigl: Securing software installations through deep graph learning. In *USENIX Security Symposium*, 2021.

[51] 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 System Security (NDSS)*, 2019.

[52] W. U. Hassan, A. Bates, and D. Marino. Tactical provenance analysis for endpoint detection and response systems. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2020.

[53] M. N. Hossain, J. Wang, R. Sekar, and S. D. Stoller. Dependence-preserving data compaction for scalable forensic analysis. In *USENIX Security Symposium*, 2018.

[54] C. Hwang and T. Lee. E-sfd: Explainable sensor fault detection in the ics anomaly detection system. *IEEE Access*, 9, 2021.

[55] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023.

[56] E. G. Ismail, A. Chahboun, and N. Raissouni. Fernet symmetric encryption method to gather mqtt e2e secure communications for iot devices. 2020.

[57] T. Isohara, K. Takemori, and A. Kubota. Kernel-based behavior analysis for android malware detection. In *2011 seventh international conference on computational intelligence and security*. IEEE, 2011.

[58] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen. Magic: Detecting advanced persistent threats via masked graph representation learning. *arXiv preprint arXiv:2310.09831*, 2023.

[59] B. Jiang, T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui, S. Iqbal, X. Han, and T. Pasquier. Orthrus: Achieving high quality of attribution in provenance-based intrusion detection systems. In *Security Symposium (USENIX Sec'25). USENIX*, 2025.

[60] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX security symposium (USENIX security 17)*, 2017.

[61] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[62] I. J. King and H. H. Huang. Euler: Detecting network lateral movement via scalable temporal link prediction. In *Network and Distributed System Security Symposium*, 2022.

[63] M. Kumar, A. K. Singh, and T. S. Kumar. Secure log storage using blockchain and cloud infrastructure. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2018.

[64] M. Laclau, L. Renou, and X. Venel. Robust communication on networks. *arXiv preprint arXiv:2007.00457*, 2020.

[65] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI*

conference on artificial intelligence, volume 29, 2015.

[66] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR, 2014.

[67] K. H. Lee, X. Zhang, and D. Xu. Loggc: garbage collecting audit log. In *CCS*, 2013.

[68] J. Li, X. Tong, J. Liu, and L. Cheng. An efficient federated learning system for network intrusion detection. *IEEE Systems Journal*, 2023.

[69] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2, 2020.

[70] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019.

[71] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12), 2018.

[72] L. Lyu, H. Yu, and Q. Yang. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[73] D. Man, F. Zeng, W. Yang, M. Yu, J. Lv, and Y. Wang. Intelligent intrusion detection based on federated learning for edge-assisted internet of things. *Security and Communication Networks*, 2021, 2021.

[74] A. A. Mansour Bahar, K. S. Ferrahi, M.-L. Messai, H. Seba, and K. Amrouche. Fedhe-graph: Federated learning with hybrid encryption on graph neural networks for advanced persistent threat detection. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, pages 1–10, 2024.

[75] E. Manzoor, S. M. Milajerdi, and L. Akoglu. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[76] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE access*, 9, 2021.

[77] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*. 2016.

[78] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[79] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. Venkatakrishnan. POIROT: Aligning attack behavior with kernel audit records for cyber threat hunting. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[80] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan. HOLMES: Real-time apt detection through correlation of suspicious information flows. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[81] K. Mukherjee, J. Wiedemeier, T. Wang, J. Wei, F. Chen, M. Kim, M. Kantarcioglu, and K. Jee. Evading {Provenance-Based}{ML} detectors with adversarial system actions. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[82] L. Muñoz-González, K. T. Co, and E. C. Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*, 2019.

[83] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, 2011.

[84] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[85] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019.

[86] R. Paccagnella, P. Datta, W. U. Hassan, A. Bates, C. Fletcher, A. Miller, and D. Tian. Custos: Practical tamper-evident auditing of operating systems using trusted execution. In *Network and distributed system security symposium*, 2020.

[87] A. O. Portillo-Dominguez and V. Ayala-Rivera. Towards an efficient log data protection in software systems through data minimization and anonymization. In *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE, 2019.

[88] L. Qu, Y. Zhou, P. P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, and D. Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[89] V. Raj, S. Suresh, and M. Phridviraj. Performance analysis of hybrid cryptographic algorithms in serverless platforms. In *International Conference on Advanced Computing and Intelligent Technologies*. Springer, 2023.

[90] M. U. Rehman, H. Ahmadi, and W. U. Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *IEEE Symposium on Security and Privacy (S&P)*, 2024.

[91] A. Riddle, K. Westfall, and A. Bates. Atlasv2: Atlas attack engagements, version 2, 2023.

[92] A. Roy Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha. Crypte: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

[93] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux. Poseidon: Privacy-preserving federated neural network learning. *arXiv preprint arXiv:2009.00349*, 2020.

[94] Y. Shen and G. Stringhini. Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *USENIX Security Symposium*, 2019.

[95] A. J. Slagell, K. Lakkaraju, and K. Luo. Flaim: A multi-level anonymization framework for computer and network logs. In *LISA*, volume 6, 2006.

[96] N. D. H. Son, H. T. Thi, P. T. Duy, and V.-H. Pham. Xfedgraph-hunter: An interpretable federated learning framework for hunting advanced persistent threat in provenance graph. In *International Conference on Information Security Practice and Experience*, pages 546–561. Springer, 2023.

[97] F. Tramer and D. Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

[98] B. Wang, Y. Chen, H. Jiang, and Z. Zhao. Ppefl: Privacy-preserving edge federated learning with local differential privacy. *IEEE Internet of Things Journal*, 2023.

[99] L. Wang, X. Shen, W. Li, Z. Li, R. Sekar, H. Liu, and Y. Chen. Incorporating gradients to rules: Towards lightweight, adaptive provenance-based intrusion detection. *arXiv preprint arXiv:2404.14720*, 2024.

[100] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter, et al. You are what you do: Hunting stealthy malware via data provenance analysis. In *Network and Distributed System Security (NDSS)*, 2020.

[101] S. Wang, Z. Wang, T. Zhou, H. Sun, X. Yin, D. Han, H. Zhang, X. Shi, and J. Yang. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17, 2022.

[102] B. Wilbert and L. Chen. Log management and retention in corporate environments. In *Proceedings of the International Conference*

on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE). The Steering Committee of The World Congress in Computer Science, Computer ..., 2012.

[103] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 13(1), 2022.

[104] B. Xia, J. Yin, J. Xu, and Y. Li. Loggan: A sequence-based generative adversarial network for anomaly detection based on system logs. In *Science of Cyber Security: Second International Conference, SciSec 2019, Nanjing, China, August 9–11, 2019, Revised Selected Papers 2*. Springer, 2019.

[105] W. Xiong, E. Legrand, O. Åberg, and R. Lagerström. Cyber security threat modeling based on the mitre enterprise att&ck matrix. *Software and Systems Modeling*, 21(1), 2022.

[106] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang. Prographer: An anomaly detection system based on provenance graph embedding. 2023.

[107] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao. {PrivateFL}: Accurate, differentially private federated learning via personalized data transformation. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[108] D. Ye, R. Yu, M. Pan, and Z. Han. Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access*, 8, 2020.

[109] X. Yi, R. Paulet, E. Bertino, X. Yi, R. Paulet, and E. Bertino. *Homomorphic encryption*. Springer, 2014.

[110] O. Zari, C. Xu, and G. Neglia. Efficient passive membership inference attack in federated learning. *arXiv preprint arXiv:2111.00430*, 2021.

[111] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *IEEE Symposium on Security and Privacy (SP)*, 2022.

[112] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[113] Y. Zhao, H. Zhou, and Z. Wan. Superfl: Privacy-preserving federated learning with efficiency and robustness. *Cryptology ePrint Archive*, 2024.

[114] Z. Zhong, W. Bao, J. Wang, X. Zhu, and X. Zhang. Flee: A hierarchical federated learning framework for distributed deep neural network over cloud, edge, and end device. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(5), 2022.

[115] T. Zhou, J. Zhang, and D. H. Tsang. Fedfa: Federated learning with feature anchors to align features and classifiers for heterogeneous data. *IEEE Transactions on Mobile Computing*, 2023.

[116] M. F. Zolkipli and A. Jantan. An approach for malware behavior identification and classification. In *2011 3rd international conference on computer research and development*, volume 1. IEEE, 2011.

Appendix A. Dataset Description

DARPA E3. The E3 dataset contains two kinds of threat actors: a nation-state actor and a common threat actor. The goal of the nation-state attacker is to steal proprietary information from the targeted company. Initially, the attacker intends to exploit the webserver hosted on FreeBSD, inject into the SSHD process, and then wait. At this point, the attacker monitors connections and network activity while residing on the FreeBSD host. Subsequently, the attacker targets and exploits the discovered hosts to exfiltrate proprietary data. The common threat attacker aims to steal Personal Identifiable Information (PII) for financial gain by deceiving targeted users into providing access to the network using spear phishing.

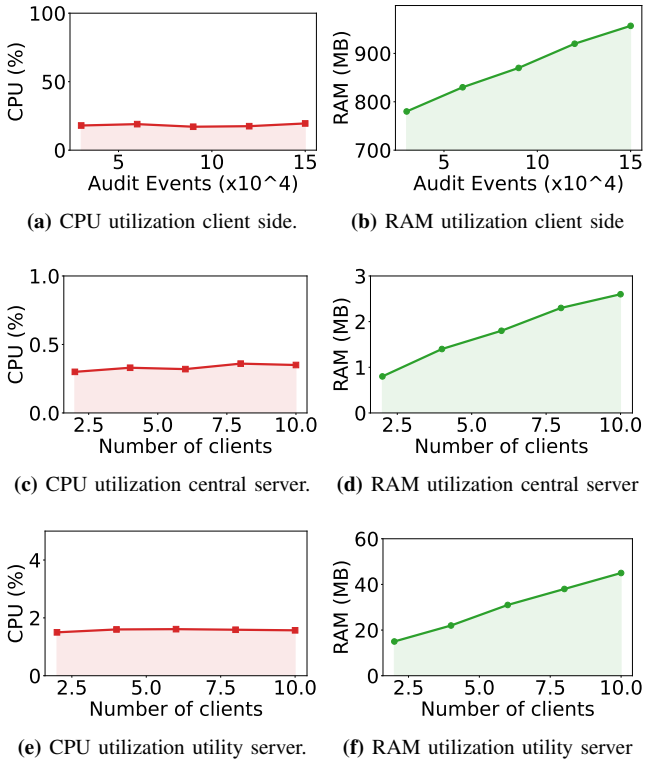
DARPA E5. The DARPA E5 dataset is more advanced than E3 in terms of attack complexity and data volume. The attackers' capabilities span the entire MITRE ATT&CK [105] adversarial lifecycle, from backdoors to exploit shellcode, from download to the execution of APT simulacra in-memory of the exploited process's memory, from fingerprinting and surveying the target to exfiltrating sensitive data. Multiple variations of APTs and attack capabilities were delivered for Windows, Ubuntu, FreeBSD, and Android. The logs from E3 and E5 are documented under various scenario names, including Cadets, Trace, Theia, and ClearScope. The attacks in E3 were conducted on a single host per scenario, whereas E5 features three hosts per scenario, presenting a multi-host threat environment.

DARPA OpTC. The OpTC dataset, another open-source resource from DARPA, encompasses a comprehensive collection of audit logs from an enterprise environment with 1,000 hosts. This dataset includes six days of benign system logs. Subsequently, attack logs span three days of system activities, featuring red team tactics such as initial compromises, privilege escalations, malicious software installations, and data exfiltration. Each attack day targeted different host machines.

Appendix B. RQ6: Resource consumption and Processing Overhead

Resource Consumption. We conducted experiments to analyze the resource consumption of the central, utility server, and client-side modules of MIRAGE. We modeled the resource utilization on a client machine using different batches of audit events of varying sizes. For the central and utility servers, we studied resource consumption by varying the number of clients to understand the demands of federated averaging and semantic vector harmonization. The results, depicted in Figure 6, indicate that MIRAGE's resource consumption is moderate. Specifically, MIRAGE can process up to 100,000 audit events simultaneously while consuming less than 900 MB of memory and utilizing less than 20% of CPU resources. This performance suggests that MIRAGE does not significantly burden the client machine, especially considering the typically low event throughput on such machines. Additionally, our analysis of the host data in the OpTC dataset shows that, on average, each client generates approximately 100,000 audit log events within a three-hour period. For the central and utility servers, the resource usage is minimal, demonstrating that our architecture is scalable and suitable for large organizations with many clients.

Processing Time Analysis. We conducted experiments to study the end-to-end processing time of our system for a client machine. For this, we used batches of audit events of various sizes, conducting end-to-end inference



(a) CPU utilization client side. **(b)** RAM utilization client side

(c) CPU utilization central server. **(d)** RAM utilization central server

(e) CPU utilization utility server. **(f)** RAM utilization utility server

Figure 6: Resource consumption of various components of MIRAGE.

with MIRAGE to measure the time taken to process these events on a client machine. The results, illustrated in Figure 7, demonstrate that MIRAGE processes events with notable efficiency. For example, it requires approximately 23 seconds to process a batch of 100,000 events. Given our previous analysis of host logs in the OpTC dataset, which indicated that each host generates an average of 100,000 events in three hours, MIRAGE can process 24 hours worth of log data on a client in merely 3 minutes. This level of efficiency ensures that our system is highly effective, preventing any potential log congestion.

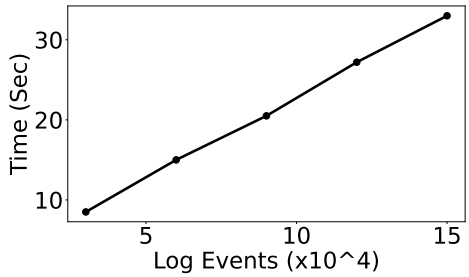


Figure 7: Processing time for various audit event sizes evaluated using OpTC dataset.

Appendix C. RQ7: Ablation study

In this ablation study, we analyze the impact of key parameters within MIRAGE. Specifically, we examine the effects of number of federated averaging rounds, the number of GNN categorized models, the anomaly threshold and differential privacy on accuracy. The effects of these parameters are discussed below:

C.1. Effectiveness of Word2vec harmonization

We evaluated the effectiveness of our Word2vec vector harmonization scheme through two experiments using the OpTC, E3 and E5 datasets. In the first experiment, each client utilized its locally trained Word2vec model to encode semantic features during the training process. In the second experiment, we harmonized the individually trained models into a central Word2vec model using the utility server architecture, as explained in Section 4. Then each client used this centralized model for generating semantic features. Table 5 presents the average results for the DARPA E3, E5 and OpTC datasets. By employing the harmonized models, we achieved significantly better detection outcomes. This is because these local models encode different information for identical tokens across hosts. Such variability leads to heterogeneity in the feature space for the GNN model, impairing the model's ability to generalize and converge effectively during the federated learning process, thereby yielding suboptimal results. However, through our novel, privacy-preserving aggregation of these semantic models, we have addressed this issue by merging the information in these models together to give the GNN more generalized input vectors.

TABLE 5: Effectiveness of Word2vec harmonization.

Dataset	Type	Prec.	Rec.	F-Score
OpTC	Local	0.58	0.97	0.73
	Harmonized	0.90	0.92	0.91
E3	Local	0.83	0.96	0.89
	Harmonized	0.96	0.99	0.97
E5	Local	0.81	0.94	0.87
	Harmonized	0.98	0.98	0.98

C.2. Effectiveness of categorized graph learning

To examine the effectiveness of our process categorization based GNN ensemble technique, we conducted experiments comparing our ensemble technique against a single model approach in a federated setting. Specifically, for the ensemble method, we designated the number of categories to be 10. This approach standardizes all processes across various hosts into 10 distinct categories, ensuring that the GNN models learn similar distributions regardless of the host. Such standardization is crucial for mitigating the impact of heterogeneous distributions and data imbalance during the federated averaging process. Table 6 reports the average results for the DARPA E3, E5 and OpTC datasets. The results indicate that utilizing categorized ensemble models yields superior performance. This improvement is attributed

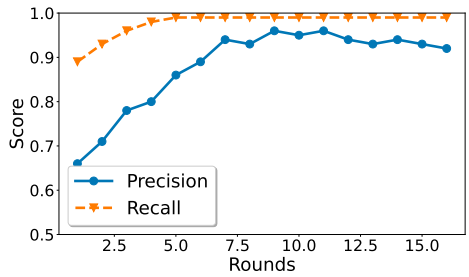


Figure 8: Federated averaging rounds vs detection performance using E3 dataset.

to each sub-model's enhanced ability to concentrate on different patterns of system activity from different clients, thereby reducing the likelihood of these patterns becoming conflated during the federated averaging process. We also studied the effect of different number of categories on detection accuracy in Appendix C.4.

TABLE 6: Effectiveness of categorization-based ensemble learning.

Dataset	Type	Prec.	Rec.	F-Score
OpTC	Single	0.86	0.91	0.88
	Ensemble	0.90	0.92	0.91
E3	Single	0.89	0.99	0.94
	Ensemble	0.96	0.99	0.97
E5	Single	0.92	0.96	0.94
	Ensemble	0.98	0.98	0.98

C.3. Effect of federated averaging rounds

We employed the DARPA E3 dataset to examine the impact of federated averaging rounds on detection performance. Our methodology involved training the model over a range of federated averaging rounds and subsequently evaluating the model's detection capabilities. The outcomes are depicted in Figure 8, which shows that detection performance improves up to a certain number of rounds before declining due to overfitting. Notably, this inflection point is also characterized by a minimal decrease in training loss, suggesting that the model has reached its learning capacity. This observation proves to be a valuable metric for determining the optimal moment to stop training, thereby preventing overfitting and ensuring optimal model performance.

C.4. Effect number of categories vs detection

We studied the impact of varying the number of categories (k) on the detection performance. Within our system entity-level GNN learning framework, k controls the creation of distinct standardized bins. These bins categorize processes across client machines. Subsequently, we employ an ensemble approach, deploying a separate GNN model for each bin. Thus, the total number of GNN models corresponds to the number of categories. The results depicted in Figure 9 indicate that an increase in k enhances precision while maintaining recall levels. This outcome arises because a single model suffices for achieving high recall by effectively distinguishing between benign and malicious patterns. Nevertheless, a single model falls short in fully generalizing across the diverse benign patterns unique to each client due to the potential blending of individual patterns during the process of federated averaging. Employing a specialized GNN strategy across various categories addresses this challenge by minimizing false alarms.

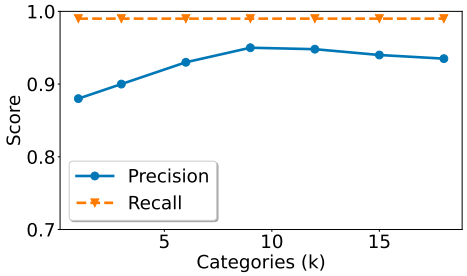


Figure 9: Effect of number of categories vs detection performance using E3 dataset.

C.5. Effect number of categories on training cost

We analyzed the impact of the number of submodels on the training costs in FL. The training cost is influenced by two primary factors: the time required to train a GNN model on a single client machine and the network overhead incurred from communicating multiple models. Our analysis of these factors revealed that each GNN model in the ensemble is consistently 13kb, a size that is determined by the model architecture and does not vary across different datasets. According to our ablation study C.4, the optimal number of categories is 10, suggesting that the impact on network overhead is minimal. Furthermore, the overall training cost remains unaffected since the total amount of data does not change; it is merely divided into different groups. This division actually enhances the training process by enabling parallel training of the models, highlighting the efficiency of our technique. Therefore, the training duration for our ensemble model is primarily limited by the time required to train the submodel managing the largest data bin.

C.6. Anomaly threshold

This hyperparameter controls the sensitivity of the MIRAGE system by controlling the number of alerts it generates. Specifically, it sets the cutoff score above which an input is flagged as anomalous. The choice of threshold directly impacts the system's precision and recall balance: a lower threshold increases recall but may lead to more false positives, whereas a higher threshold boosts precision at the cost of missed detections. To analyze this trade-off, we evaluate the detection performance on the E5 dataset by varying the anomaly threshold and plotting the resulting ROC-AUC curves. As illustrated in Figure 10, MIRAGE achieves an AUC of 0.97, indicating high robustness and effective separation between benign and anomalous instances across the tested threshold range.

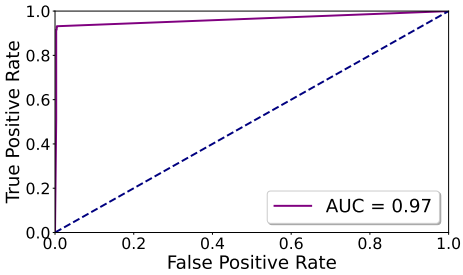


Figure 10: Anomaly threshold effect.

Appendix D.
RQ8: Effect of differential privacy on accuracy

In Section 6, we analyze how our system design offers strong guarantees against model inference attacks. Here we discuss another alternative technique for preserving privacy through the use of Differential Privacy (DP). It can be integrated with federated learning to protect against inference attacks [72, 85, 110], though it comes at the cost of detection accuracy.

Differential privacy safeguards the model by injecting a controlled amount of Gaussian noise into its parameters, thereby concealing the influence of any individual data point. In our work, we adopt a node-level DP strategy, which ensures that each node's features and labels are protected when noise is added to the gradient updates. As a result, individual node contributions remain obscured in the aggregated model parameters, reducing the likelihood of identifying specific nodes or their features. We conduct experiments to examine how varying levels of differential privacy noise affect the detection performance of MIRAGE. The noise is adjusted based on a privacy budget defined by ϵ. This noise is applied to local GNN model updates before they are aggregated at the central server during federated averaging, as described in Section 4.

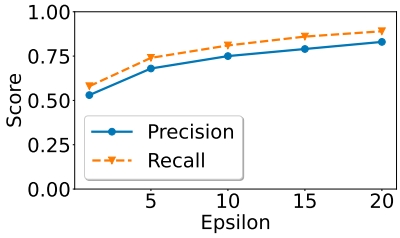


Figure 11: Effect of differential privacy noise on detection using E3 dataset. Note that we observed similar results on the other datasets.

The privacy budget ϵ plays a pivotal role. Lower values of ϵ increase the noise injected into the model, which enhances privacy at the expense of detection performance. Conversely, higher values of ϵ inject less noise, improving detection accuracy but offering weaker privacy guarantees. By tuning ϵ during training, we explore the balance between privacy and detection effectiveness across various settings. As shown in Figure 11, increasing the noise level (i.e., reducing ϵ) degrades the model's utility, thereby providing more privacy at the cost of lower accuracy.

Hence, although DP is a valid solution for protecting privacy, the accuracy deterioration that comes with it reduces its utility in the domain of intrusion detection, where high detection performance is needed. Due to these reasons, we do not use DP in MIRAGE and instead design a dual-server architecture to offer privacy guarantees without adding noise to the model updates.

Appendix E.
Comparison of encryption methods

We chose Fernet symmetric key encryption for its ability to efficiently handle large datasets while ensuring confidentiality and integrity. Fernet employs AES-based symmetric encryption along with an HMAC for authenticity. Because the HMAC relies on efficient hash functions rather than expensive asymmetric operations [89], Fernet is computationally more efficient for processing large volumes of data.

In contrast, asymmetric methods like RSA which use public-private key pairs and are commonly employed for key exchange are significantly slower and therefore unsuitable for encrypting large datasets [78]. Their computational overhead makes them impractical for scenarios involving substantial data, such as Word2vec models with thousands of tokens.

This efficiency is crucial in MIRAGE, where we encrypt Word2vec models that can contain a large number of tokens. In our experiments, encrypting and decrypting a Word2vec model with 1000 tokens took Fernet only 0.11 ms, compared to 0.64 ms with RSA, making Fernet nearly six times faster.

After encryption, the utility server harmonizes these encrypted models by performing mean averaging on the vectors of overlapping tokens. It identifies such overlaps, averages their vectors, and updates the model accordingly before returning it to the clients.

Our design ensures privacy without resorting to Homomorphic Encryption (HE). Although HE allows computations directly on encrypted data, it imposes a significant performance penalty [83]. Its mathematically complex operations reduce throughput and scalability. By avoiding HE, we maintain strong privacy protections without incurring the prohibitive computational costs that would impede efficient real-time model updates.

Appendix F.
State Explosion in Massive Networks

Scaling FL to larger enterprise networks introduces potential state explosion issues, primarily due to the increasing complexity in managing and integrating updates from a growing number of clients. As more clients participate, each contributing their local model updates, the volume of data to be processed and aggregated can increase significantly. This escalation can strain communication channels, leading to higher bandwidth requirements and increased latency. Furthermore, the aggregation process itself becomes more

computationally demanding as the number of updates grows. MIRAGE addresses these challenges by utilizing an ensemble learning approach and categorizing system activities, which simplifies the aggregation by processing more uniform data segments. Moreover, our system uses models with small network overhead to prevent bandwidth challenges. To further enhance efficiency, additional techniques such as model compression [30], selective update aggregation [108] and the implementation of more robust communication protocols [64] can significantly improve handling large volumes of updates, thus preventing state explosion and maintaining system performance in large-scale federated learning environments.