Seeing Without Seeing: A Privacy-Aware Approach to System Intrusion Detection

ABSTRACT

Enterprises increasingly rely on Intrusion Detection Systems (IDS) to detect malicious threats through log analysis. However, centralizing these logs, which often contain sensitive data (e.g., URLs, file activities), raises privacy concerns. In this paper, we propose TrustWatch that leverages a novel adaptation of Federated Learning (FL) and graph representation learning to enable local model training on system logs at client machines without sending raw logs to a central server, thus promoting decentralization and privacy in IDS. FL presents challenges in IDS due to varying data distribution and heterogeneity among clients, which impacts the generalization of the global model. To mitigate this, we introduce a novel ensemble learning approach and a process entity categorization scheme, where each submodel specializes in distinct system activity patterns across clients, preventing conflation during model aggregation. Semantically rich feature vectors are crucial for high detection accuracy; however, semantic encoders, such as Word2Vec complicate private aggregation on a central server in the FL process. To resolve this, we developed a Word2Vec harmonization framework within a multi-server architecture that securely aggregates semantic attributes. One server generates encryption keys for client-level log attributes, while another performs computations on encrypted data without revealing sensitive tokens. Extensive evaluations on DARPA E3, E5, and OpTC datasets show that our system achieves state-of-the-art detection accuracy while being highly scalable and privacy-preserving.

CCS CONCEPTS

• **Security and privacy → Intrusion/anomaly detection**; **Machine learning**; **Operating systems security**.

KEYWORDS

Data provenance; Federated learning; Intrusion detection

ACM Reference Format:

. 2025. Seeing Without Seeing: A Privacy-Aware Approach to System Intrusion Detection. In . ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Intrusion Detection Systems (IDS) are crucial for countering Advanced Persistent Threats (APTs) in enterprises, as evidenced by major attacks, such as Solar Winds [10] and NotPetya [9]. Given their stealth and persistence, many enterprises rely on Managed Security Service Providers (MSSPs) to improve their defenses. A survey [7] involving over 5,000 IT professionals reported that about 75% of companies use MSSPs. These providers integrate their security systems with clients' systems to manage system logs, typically configuring the clients' systems to transmit system logs to the cloud for analysis. Figure 1 illustrates this MSSP operational architecture.

Recently, Provenance-based IDS (PIDS) [33, 52, 53, 65, 88, 102, 110, 111, 116, 121] have emerged as a highly effective means of enhancing intrusion detection. These systems leverage the extensive contextual information provided by system logs to bolster their detection capabilities. PIDS operates by converting system logs into provenance graphs, which are then analyzed using machine learning techniques, such as Graph Neural Networks (GNN), to identify and learn patterns of benign activity. By continuously monitoring these patterns, PIDS can detect deviations that may indicate potential security threats. Upon detecting such anomalous graph patterns, the PIDS generates alerts to prompt further investigation.

1.1 Limitations of Existing PIDS

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

Privacy Risks and Centralized Dependency. Current PIDS rely on a centralized infrastructure where client machines transmit their logs to a central server. While this centralization is necessary for aggregating large datasets to effectively model and understand benign behaviors using advanced deep learning techniques, it raises significant privacy concerns. Logs often contain sensitive information, such as URLs visited, IP addresses, and application usage, which can be exposed in centralized systems. These privacy risks are not merely theoretical; they have been highlighted in a recent Datadog report [1]. Additionally, training models on data from a single machine is insufficient, as shown by our experiments with Flash on the DARPA OPTC dataset [4], where performance dropped by 40% in F-score when using single-host data compared to multi-host data.

Scalability and Operational Inefficiencies Centralized PIDS face significant challenges related to both network overhead and scalability. Transmitting large volumes of logs over the network for intrusion detection imposes substantial costs on users and organizations. Modern systems can generate gigabytes of logs daily [59, 61]. Our analysis of PIDS, such as Flash and Kairos, using the OpTC dataset as detailed in Section 6.4, highlights this issue. For organizations similar in scale to those in the OpTC dataset, daily log volumes can reach 1000 GB. This leads to considerable network expenses and difficulties for users with limited bandwidth who struggle to upload such large amounts of logs efficiently. As the number of hosts within an organization increases, centralized PIDS often experience log congestion, creating bottlenecks that slow down intrusion detection. Centralization also results in significant disk storage

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

https://doi.org/10.1145/nnnnnnn.nnnnnnn

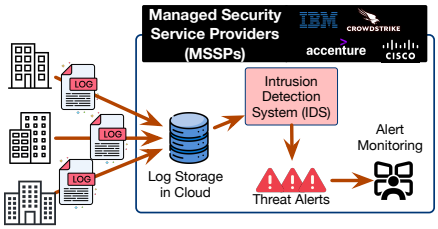


Figure 1: The MSSP architecture for intrusion detection, which is vulnerable to privacy leakage. The plain-text system logs are first collected in a centralized storage managed by the MSSP. Then these system logs are analyzed for intrusions.

overhead, requiring continuous resource allocation to manage the growing data. Our evaluations of FLASH and KAIROS show that these systems would face severe log congestion in environments comparable to the OpTC dataset. FLASH would require 27.7 hours and KAIROS 56.6 hours to process a single day's logs, highlighting the lack of scalability in current centralized PIDS solutions.

1.2 Our Approach & Contributions

We introduce TRUSTWATCH, a novel privacy-preserving PIDS that integrates provenance graph representation learning with Federated Learning (FL) to address the challenges in applying FL to PIDS. Prior work [111] has shown the effectiveness of graph representation learning on provenance graphs for threat detection, as opposed to applying machine learning techniques [35, 46] directly on system logs [39, 114]. Building on this insight, TRUSTWATCH leverages Federated Provenance Graph Learning, combining FL's decentralized architecture with powerful graph representation learning to capture the intricate relationships among system entities. *To the best of our knowledge, TRUSTWATCH is the first system to build a privacy-preserving PIDS and effectively solve these challenges.* In TRUSTWATCH, client logs remain local, ensuring that sensitive data never leaves the client's environment. Each client independently trains GNN models, which are aggregated on a central server with models from other clients. This approach captures diverse activity patterns without transmitting actual logs, minimizing network overhead. Only model updates—mere kilobytes per client—are transmitted, greatly reducing the network burden. All major computations, including training, take place locally on client machines, leveraging their resources for real-time threat detection. This decentralized approach allows TRUSTWATCH to scale effectively as more hosts are added, with each utilizing its own computing power and storage.

However, applying FL to PIDS introduces several significant challenges, as summarized below:

C1 Data Imbalance & Heterogeneity Among Clients. Variations in data distribution and volume across clients, often non-IID [122], pose challenges in training a global model GNN_{Global} that performs uniformly well. Heterogeneous data resulting from different client applications can lead to suboptimal performance [100], and data imbalances can cause models to be biased towards clients with more extensive datasets, potentially overlooking unique patterns in less-represented clients [41]. This issue is critical for PIDS, as a biased GNN_{Global} may result in high false alarms, undermining detection accuracy.

C2 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 [33], 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 local Graph Neural Network models GNN_i and complicates federated averaging for the global Graph Neural Network model $\text{GNN}_{\text{Global}}$, leading to suboptimal anomaly detection performance, as detailed in [123].

C3 Temporal Misalignment. Aggregating temporal graph models from clients with temporally misaligned data challenges the creation of a cohesive global model. Systems like KAIROS [33] employ temporal graph neural networks (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 trained on diverse temporal datasets. This results in a loss of temporal information and hinders the formation of a cohesive global model GNN_{Global}.

To address C1, we have designed a novel ensemble learning framework. In this framework, each submodel is trained to specialize in learning system activities associated with specific process entities, which are standardized across all clients. This standardization is facilitated by a sophisticated categorization scheme enabled by our dual-server architecture. Through this system, process entities from all clients are organized into K privacy-preserving bins. Each client then aligns its process nodes with these bins, constructs a provenance subgraph for each bin, and trains a GNN model on these graphs. The models from all clients are then aggregated into K model pairs, forming a comprehensive global ensemble model set. This strategic approach ensures that models with similar data distributions are merged effectively, thereby maintaining the integrity of unique activity patterns across diverse client environments. We compared our methods with existing solutions such as FedProx [77] and FedOpt [22] for addressing heterogeneity in FL settings and found that our techniques outperform these existing solutions, as detailed in Section 6.3. To tackle C2, 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 to achieve a unified, privacy-preserving vector representation. This method ensures that sensitive data remains protected while facilitating accurate and consistent semantic encoding across different clients. To avoid the problem of C3 arising from the use of temporal graph networks, TRUSTWATCH employs an inductive graph neural network model [50], which has been shown by prior work [102, 111, 121] to offer good performance and is less affected by temporal dependencies. This approach mitigates the issues of temporal misalignment that arise in federated learning for intrusion detection settings.

We evaluated TRUSTWATCH using open-source datasets from DARPA, including E3 [2], E5 [3], and OpTC [16], which cover a

wide range of attack scenarios and system behaviors. Since existing state-of-the-art PIDS already achieve near-perfect detection accuracy, the primary goal of TrustWatch is to demonstrate that a privacy-preserving PIDS can be built while maintaining similar performance. Our results show that TrustWatch achieves detection performance with an average precision of 96% and recall of 97%, matching the accuracy of existing PIDS. What distinguishes TrustWatch is its ability to deliver this accuracy while significantly reducing privacy risks and improving scalability in enterprise settings. Experiments were conducted to investigate the effectiveness of Word2Vec harmonization and categorized ensemble learning in addressing the data heterogeneity issue. These methods were found to significantly enhance detection performance compared to the vanilla approach.

TrustWatch achieves a 170-fold reduction in network communication costs compared to existing centralized systems. Moreover, the decentralized nature of TrustWatch allows for much faster inference times, bounded only by the client with the most data, completing the full OpTC dataset in a few minutes, whereas existing PIDS, such as Flash and Kairos, require several hours. We also provide a detailed analysis of the privacy protection offered by our system in Section 8. Additionally, we present a comprehensive analysis of our system's resilience against various adversarial attacks in Section 7. Furthermore, we provide an extensive ablation study, comparison of our approach to alternate privacy techniques such as differential privacy and benchmark of various encryption methods for use in our system in Appendix D.

2 RELATED WORK

Machine Learning based PIDS. Many existing systems leverage machine learning techniques for threat detection. ProvDetector [110] utilizes the Doc2Vec [73] model to encode attack paths in provenance graphs into embeddings for outlier detection. Attack2Vec [106] employs a temporally aware word-to-embedding encoding scheme to identify attack entities. In the domain of network anomaly detection, DeepAid [51] utilizes deep neural networks to differentiate anomalous traffic. ProGrapher [116] generates provenance graph embeddings for anomalous graph identification by integrating Graph2Vec [96] and TextRCNN [72] models. StreamSpot [88], another graph-level detector, constructs benign models using various graph features and detects anomalies through clustering algorithms. Furthermore, Unicorn [52] employs graph kernels for graph-level threat detection. Other systems [13, 14, 49, 89] also utilize diverse embedding generation techniques. Additionally, studies such as [30, 64, 124] focus on malware detection. Techniques like DeepLog [39] directly process logs using recurrent neural networks. SIGL [53] specifically targets the detection of malicious software installations, while Euler [68] employs both GNN and RNN models to detect lateral movements. MAGIC [65] uses masked graph representation learning for detecting system threats. In comparison to all these IDSes, TrustWatch is the first to utilize the principles of federated learning and address the associated challenges to achieve privacy-preserving and scalable threat detection.

DisDet [38] detects APTs using Hierarchical System Event Trees (HST) in a decentralized manner. DisDet reduces network costs by summarizing system events in the form of HST but compromises privacy because HSTs must be sent to the server in plain text. Additionally, DisDet identifies anomalies as events simply missing from the benign HST, a method prone to false alarms in dynamically changing system environments. As more new benign processes are added to the system, they will not be present in existing HSTs and would be mistakenly raised as alarms.

Rules-based IDS. Another category of PIDS concentrates on utilizing predefined rules to detect malicious entities. Key examples include Holmes [93], Rapsheet [54], and Poirot [92]. These approaches leverage insights from APTs to formulate their rule bases. In comparison to PIDS systems based on machine learning, rule-based PIDSes tend to generate fewer false positives. Nevertheless, a significant limitation of these systems is their inability to identify threats featuring novel attack signatures. Additionally, they often necessitate the expertise of skilled security professionals for the development of their rule sets.

Federating Learning in Threat Detection. Few PIDS currently utilize federated learning, with most research focusing on Network Intrusion Detection. Notable works include [87], which proposes FL for IoT threat detection, and [43], which introduces a differentially private detection system for industrial IoT. [76] presents an efficient network intrusion detection system, while [48] explores mitigating the impact of non-IID data on FL for intrusion detection. TrustWatch addresses these challenges using a categorized GNN learning framework and secure semantic vector harmonization. Lastly, [29] describes another FL-based system for IoT intrusion detection. Unlike these approaches, TrustWatch uniquely integrates federated learning with graph-based threat detection, enhancing both accuracy and privacy in diverse enterprise environments.

Privacy-Enhancing Cryptographic Techniques. Multi-Party Computation (MPC) [36] and Fully Homomorphic Encryption (FHE) [21] are two techniques for enhancing privacy in cryptographic systems. MPC enables a group of parties to compute a function over their inputs while keeping those inputs private. FHE allows for computations on encrypted data. However, these techniques can introduce additional system complexity and scalability issues [23, 40, 44] as the interaction required between parties can lead to delays. In contrast, TrustWatch uses a simple and scalable encryption technique for averaging semantic embeddings. We provide a privacy analysis of this technique in section 8.

Privacy-preserving FL. PrivateFL [117] addresses the challenge of heterogeneity introduced by differential privacy (DP) in FL systems. It introduces a personalized data transformation technique to apply DP to model updates in order to prevent inference attacks at the central server. Similarly, [17] utilizes FL and DP for the privacy-preserving detection of browser fingerprinting. [37] introduces a secure federated averaging technique using homomorphic encryption. Poseidon [105] presents a privacy-preserving neural network training scheme that employs multiparty cryptography to preserve the confidentiality of training data under a passive-adversary model. Finally, Ppefl [108] proposes a privacy-preserving FL framework based on local DP, addressing the issues of high privacy budget and poor model performance in FL scenarios. In comparison to these approaches, TrustWatch introduces a robust multi-model server architecture that is privacy-preserving and is resistant to inference

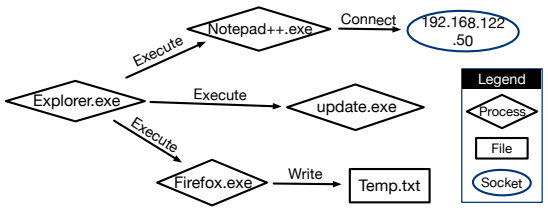


Figure 2: Data provenance graph example.

attacks without the use of DP. This prevents TrustWatch's performance from deteriorating by the noise introduced by DP while offering strong privacy guarantees.

3 BACKGROUND

Data Provenance Graphs. System logs are crucial for recording system activities, detailing interactions among entities such as processes, files, and sockets. Operating systems like Windows, Linux, and FreeBSD offer built-in mechanisms to gather these logs, such as Event Tracing for Windows (ETW)[6], the Linux Audit system[8], and DTrace [5]. Each log entry characterizes a system event, identifying a subject and object entity, along with contextual attributes like process names, file names, and socket IP addresses. Data provenance models the causal links among system entities by using logs to construct a dependency graph that captures event relationships. In this graph, nodes represent entities like processes, files, and sockets, while edges represent system syscalls. This provenance graph is essential for analyzing system activities and identifying anomalous or malicious entities. Figure 2 illustrates a provenance graph where *Explorer.exe* executes processes like *Notepad.exe* and *Firefox.exe*, which then interact with other file and socket nodes.

Federated Learning. Federated learning (FL) is a machine learning approach that enables a model to be trained across multiple decentralized devices or servers holding local data samples, without exchanging them. The mathematical formulation of FL involves aggregating locally computed updates to a global model, rather than directly sharing raw data. In a federated learning setup, consider a global model parameterized by θ. The goal is to minimize some loss function $L(\theta)$ that measures the discrepancy between the model's predictions and the actual data across all devices. However, instead of pooling all the data together, each device i computes an update $\Delta\theta_i$ based on its local data and a local loss function $L_i(\theta)$. Mathematically, the update rule on device i for a learning rate η is given by $\Delta\theta_i = -\eta\nabla L_i(\theta)$, where $\nabla L_i(\theta)$ is the gradient of the loss function with respect to the model parameters for the local data. After computing $\Delta\theta_i$, each device sends this update to a central server. The central server aggregates these updates from all participating devices to update the global model. The aggregation is an average function: $\Delta\theta_{global} = \frac{1}{N}\sum_{i=1}^{N}\Delta\theta_i$, where N is the number of devices. This aggregated update is then used to update the global model parameters: $\theta_{new} = \theta + \Delta\theta_{global}$. This process iterates, with the updated global model being sent back to the devices for further local updates, until convergence. This iterative process ensures that the global model learns from all devices while keeping the data localized, thus addressing privacy and bandwidth concerns.

4 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 [76, 87]. We also consider the possibility of a curious central server attempting membership inference attacks by utilizing the model weights.

Similar to other works in cryptography and federated learning [104, 113], we assume that the utility server is trusted and that there is no collusion between the central and utility servers, meaning they do not exchange information. To further ensure non-collusion, the utility server can be deployed within a trusted execution environment (TEE) [90] or monitored by a trusted mediator overseeing communications between the servers [15]. Alternatively, since the utility server's role is limited to processing encrypted data, it can be implemented as a secure cloud compute instance [11] managed by the organization, while the MSSP manages the central server. 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.

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). In line with prior studies on data provenance [25, 54–57, 61, 62, 70, 74, 79, 82–84, 86, 92, 109], 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. [98] and the Hardlog system [12]. Similar to other PIDS works [33, 102, 110, 111, 116], we assume the absence of any attack activities during the training phase which makes model poisoning attacks out of the scope of this work.

5 DESIGN

We define all mathematical notations for our system in Table 1.

5.1 Problem Formulation & Overview

Each client machine in the set C locally maintains its logs to preserve privacy, ensuring that raw logs do not leave the machine. Our objective is to detect anomalous nodes within the client's provenance graphs, $\{G_{c_1}, G_{c_2}, \ldots, G_{c_N}\}$, generated from these logs. To this end, we propose training a set of global GNN models $\mathcal{G}_{\text{global}}$ to model the benign behavior exhibited across all client logs without centralizing the log data.

Additionally, we aim to develop a global semantic encoder $\mathcal{E}_{\text{sem}}$, capable of converting contextual attributes into vector space, thereby generating feature vectors for training our GNN models. Using

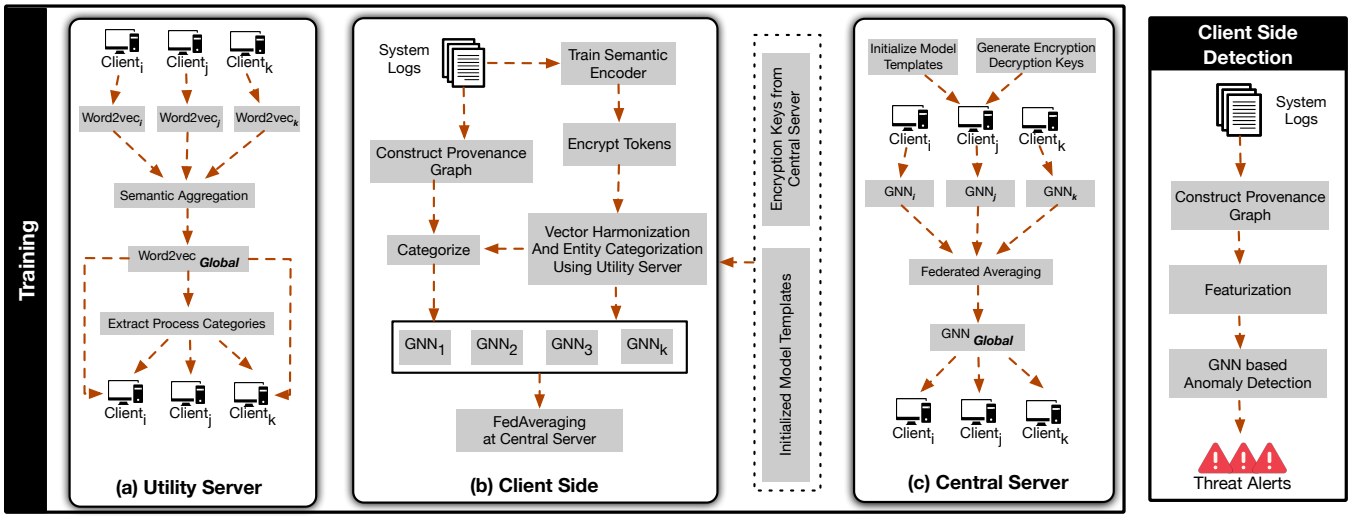


Figure 3: High-level architecture of TrustWatch. In the training phase, our system builds local provenance graphs for each client and trains an ensemble of GNN models. Prior to this, we aggregate semantic models contextually for feature encoding. The local GNN models participate in federated learning to develop a global GNN model, which is then utilized for anomaly detection. Wajih: Use key notations in this figure.

Table 1: Key Notations

Notation	Definition
N	Total number of clients in federated learning.
k	Number of categories for process entities.
C	Set of client machines $\{Client_1, Client_2, \ldots, Client_N\}$.
$G_{c_i} = (\mathcal{V}_i, \mathcal{E}_i)$	Provenance graph for client i, with nodes $\mathcal{V}_i$ and edges $\mathcal{E}_i$.
$\mathcal{G}_{\text{global}}$	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 C_j.
y_v	True label of node v.
$\hat{y}_v^j$	Predicted label of node v by submodel j.
$\mathcal{E}_{\text{sem}}$	Global semantic encoder converting contextual attributes into vector space.
$\mathcal{L}^{(r)}$	Loss function value after round r.

$\mathcal{G}_{\text{global}}$, we aim to identify anomalous nodes in clients provenance graphs G_{c_i}. These anomalies could be potential threats as their system activity significantly differs from benign patterns.

TrustWatch comprises five key modules, starting with the *Provenance Graph Constructor* module on each client machine, which transforms logs into a provenance graph. While our approach to constructing provenance graphs builds on established methods [26, 55, 58, 61, 75, 85], detailed information is provided in Appendix A. The *Semantic Featurization & Harmonization* module (Section 5.2) trains word2vec models to encode semantic attributes and consolidates individual client models into a cohesive global model using a trusted utility server and encryption techniques. The *Process Entity Categorization Module* (Section 5.3) standardizes system entities across clients, ensuring uniformity in GNN model training. The *Federated Provenance Graph Learning Module* (Section 5.4) then trains GNN models on each client machine using harmonized semantic features, with the models aggregated on a central server through federated averaging. Finally, the *Anomaly Detection Module* (Section 5.5) applies the unified global models for anomaly detection on each client machine. Figure 3 illustrates the comprehensive architecture of TrustWatch, with further details in subsequent sections.

5.2 Semantic Featurization & Harmonization

This module processes the provenance graph generated from system logs by transforming node attributes into feature vectors for the graph learning phase. Existing systems, such as Flash [102], have demonstrated the effectiveness of utilizing semantic attributes of nodes to enhance detection performance. System logs are rich in attributes related to various entities, including process names and file paths. These attributes must first be converted into vector space to serve as node features for our GNN model. For this encoding, we employ the Word2Vec semantic encoder. The Word2Vec model processes different attributes for each type of node: for process nodes, it considers the process name and command line arguments; for file nodes, the file path; for socket nodes, the network IP address and port; and for module nodes, the module name. We transform the subgraph for each node into a document by combining its semantic attributes with the types of causal events, such as system calls, involving its neighbors. These documents are then transformed into fixed-length vectors using a Word2Vec model trained on non-malicious system logs. This approach effectively captures the semantic relationships between terms, producing dense embeddings that enhance the subsequent learning of graph representations.

Overlapping Tokens. 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 $Word2vec_{Global}$ 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 downstream federated averaging technique by maintaining uniformity in the input space for the GNN models across all clients.

Non Overlapping Tokens. 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 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 5.4, where each submodel specializes in distinct system patterns across clients, enhancing model precision as shown in Section 6.

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. This finding aligns with related work [102], 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 encoder, mapping them to the same embedding space. In contrast, a federated setting, where each client trains its own encoder, 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. The Word2Vec model functions as a key-value store, with vocabulary tokens as keys, k, and their corresponding vector representations as values, v_k. To combine these models, we calculate the average vector of overlapping tokens from all client machines, creating a central model. The mathematical representation for averaging vectors of a token k across N clients is given by $\bar{v}_k = \frac{1}{N}\sum_{i=1}^{N} v_{k,i}$ where $\bar{v}_k$ is the averaged vector for token k, and $v_{k,i}$ is the vector representation of token k from the i-th client model.

However, 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 [27, 63] 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: $D(v_k^{'}) = v_k$. We present experimental studies in Section D that compare the efficiency of Fernet encryption with other methods.

Algorithm 1: Privacy preserving harmonization of Word2Vec models.

Inputs : Client Word2Vec models $\{Word2vec_1, Word2vec_2, \ldots, Word2vec_k\}$.
Output: Harmonized Word2Vec model $Word2vec'_{Global}$ 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 ← EncryptModelTokens(Word2vec_i, E)   /* Encrypt tokens
      using E. */
    Send Wor2vec_i to Utility Server
end
/* Utility server merges encrypted models.                              */
TokenVectors ← InitializeEmptyDictionary()
TokenCounts ← InitializeEmptyDictionary()
foreach encrypted model Word2vec_i do
    foreach token t in Word2vec_i do
        Vector ← Word2vec_i[t]
        if TokenVectors.HasKey(t) then
            TokenVectors[t] ← TokenVectors[t] + Vector
            TokenCounts[t] ← TokenCounts[t] + 1
        else
            TokenVectors[t] ← Vector
            TokenCounts[t] ← 1
        end
    end
end
/* Average the vectors for overlapping tokens.                          */
foreach token t in TokenVectors.Keys() do
    TokenVectors[t] ← TokenVectors[t]/TokenCounts[t]
end
Word2vec_Global ← NewModel(TokenVectors, EncryptedTokens)
  /* Constructing a new harmonized model. */
foreach client C_i do
    Send Word2vec_Global to C_i
end
return Harmonized model Word2vec_Global has been dispatched to all clients.
```

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 explained in detail in Algorithm 1.

5.3 Process Entity Categorization

Our experimental results, presented in Table 3, indicate that a single $\mathcal{G}_{\text{global}}$ model cannot achieve good detection performance in an FL setting due to the diverse and heterogeneous distributions of clients' data in the set C. 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 employs a central utility server that plays a pivotal role in the categorization process. Clients transmit lists of encrypted process names to the utility server to ensure privacy. The server aggregates these lists into a global list of process names $\mathcal{P}_{\text{global}}$ and then applies the categorization map $\psi(p)$ to randomly

divide them into k global categories, where k is a predefined hyperparameter. Although this division is random, it is performed only once at the global level. Consequently, all occurrences of a given process name are mapped consistently across every client, such that $\psi(p) = j$ assigns process p to category j.

To illustrate this, consider two clients: A and B:

- A has a set of processes {`chrome`, `ssh`, `mysql`}.
- B has a set of processes {`chrome`, `firefox`, `ssh`, `mysql`}.

Both clients send encrypted lists of these process names to the utility server, which merges them into a single global list of unique names {`chrome`, `ssh`, `mysql`, `firefox`}. The server then applies ψ to randomly assign each name to one of the k categories, where k is a predefined hyperparameter. For instance, ψ(`chrome`) and ψ(`mysql`) might map to category 1, while ψ(`ssh`) and ψ(`firefox`) map to category 2.

This categorization is sent back to both clients. Now all occurrences of `chrome` and `mysql` fall under category 1 for both clients, while `ssh` and `firefox` fall under category 2. In this way, each category holds similar processes across all clients, guaranteeing consistency even though the initial split was random.

We conducted experiments to explore the impact of k on detection accuracy and FL training cost, as detailed in Appendix D.

The clients use the processes within each bin to generate provenance subgraphs G_{c_i}. The neighborhood hop length of these subgraphs is set to match the number of graph convolution layers in the $\mathcal{G}_{\text{global}}$. This ensures that the $\mathcal{G}_{\text{global}}$ has complete neighborhood information for each node and that no interconnections are lost due to the categorization process. In our experiments, we set the hop length to two, which has been shown by prior works [102, 111] to offer an optimal balance between efficiency and detection accuracy. These subgraphs serve as training data for an ensemble of $\mathcal{G}_{\text{global}}$ models. Each submodel in the ensemble is trained on the subgraphs corresponding to its designated category and participates in federated averaging.

This approach ensures a balanced segmentation of data across clients, maintaining consistency in the training datasets for each $\mathcal{G}_{\text{global}}$ model before federated averaging. By dividing the overall task into sub-tasks (sub-models) and assigning them to different subsets of processes, the influence of clients with large datasets is distributed across multiple models. This prevents any single client from disproportionately affecting the outcome of the federated learning system. Each sub-model specializes in a different aspect of the data, capturing unique patterns and distributions. This specialization promotes more balanced contributions across the ensemble, enhancing the robustness and performance of the overall system.

5.4 FL with Categorized Provenance Subgraphs

Advanced persistent threats involve multiple causally linked attack steps across various system entities, highlighting the need to capture and model the interactions among these entities for effective detection. Analyzing each system event in isolation does not allow us to capture these interactions properly, as observed in existing log-level systems [39, 78, 114]; hence, provenance graphs, G_{c_i}, are being used to effectively model the interaction of system entities. Moreover, Graph Representation Learning is used to learn the patterns present in these graphs, as shown in related works [33, 65, 102]. We integrate federated learning with graph representation learning to bring privacy and decentralization to intrusion detection while maintaining the strong detection performance offered by the provenance graph learning technique.

Our approach includes a central server responsible for initializing the global GNN models, $\mathcal{G}_{\text{global}}$, with random weights, $w_j^{(0)}$, which are then sent to all clients in C. These clients use their local process subgraphs, G_{c_i}, and semantic feature vectors to train the GNN models in an unsupervised way, following a training method similar to Flash. The objective of the GNN model is to classify each node v into its corresponding type, yielding predictions $\hat{y}_v^j$. The server then applies the federated averaging algorithm to merge the GNN models into a set of global models, $\mathcal{G}_{\text{global}}$, based on the process entity categories on which they were trained. This ensures that models with similar distributions are combined together to address the data heterogeneity problem. Specifically, the server aggregates parameters from N client models to update the global model as $\bar{w} = \frac{1}{N}\sum_{i=1}^{N} w_i$.

The federated averaging process is repeated for a set number of rounds R, and concludes when there is no further reduction in the training loss, $\mathcal{L}^{(r)}$. Algorithm 2 shows the training and aggregation process of GNN for a given process category and one round of FL.

5.5 GNN-based Anomaly Detection

TrustWatch employs a standard 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. In operational 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.

TrustWatch performs threat detection in a decentralized manner on client's provenance graphs ($\{G_{c_1}, G_{c_2}, \ldots, G_{c_P}\}$) in an organization. For a given provenance graph on a client machine containing nodes V and edges E, TrustWatch uses the global $\{GNN_1, GNN_2, \ldots, GNN_k\}$ GNN models trained using federated learning. Each submodel performs inference on 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 y_v^i. A node v is identified as an anomaly if it is misclassified by all submodels, indicating that none of the submodels recognize the neighborhood structure or features displayed by this node.

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.

6 EVALUATION

Wajih: Be consistent with datasets in each research question. Sometimes you use OpTC and sometimes you use E3 and you don't even

Table 2: Comparison of TrustWatch against Flash and Kairos. Prec.: Precision; Rec.: Recall; Flash has slightly better performance than our system, but TrustWatch preserves user privacy and achieves high scalability through decentralization, unlike Flash and Kairos.

Datasets	TrustWatch				Flash [102]				Kairos [33]			
	Prec.	Rec.	F-Score	TP/ FP/ FN/ TN	Prec.	Rec.	F-Score	TP/ FP/ FN/ TN	Prec.	Rec.	F-Score	TP/ FP/ FN/ TN
E3-CADETS	0.97	0.99	0.98	12846/ 389/ 6/ 706,577	0.99	0.99	0.99	12845/ 188/ 7/ 706,778	0.80	1.00	0.89	4/ 1/ 0/ 174
E3-TRACE	0.95	0.99	0.97	67357/ 3196/ 26/ 2,412,811	0.99	0.99	0.99	67357/ 408/ 26/ 2,415,599	-	-	-	-
E3-THEIA	0.95	0.99	0.97	25313/ 1211/ 49/ 3,504,115	0.99	0.99	0.99	25313/ 301/ 49/ 3,505,025	0.91	1.00	0.95	10/ 1/ 0/ 216
OpTC	0.90	0.92	0.91	596/ 65/ 54/ 1,287,290	0.93	0.92	0.93	600/ 41/ 50/ 1,287,314	0.84	1.00	0.91	32/ 6/ 0/ 1210
E5-CADETS	0.99	0.92	0.95	214687/ 1049/ 19570/ 1075796	0.99	0.93	0.96	216716/ 1200/ 17541/ 1075645	1.00	1.00	1.00	16/ 0/ 0/ 238
E5-THEIA	0.99	0.98	0.99	224944/ 34/ 4213/ 1097566	0.99	0.98	0.99	225150/ 100/ 4007/ 1097500	0.67	1.00	0.80	2/ 1/ 0/ 173
E5-ClearScope	0.99	0.97	0.99	13679/ 3/ 388/ 83345	0.99	0.96	0.98	13533/ 40/ 534/ 83308	0.67	1.00	0.80	10/ 5/ 0/ 217

explain why you excluded the other datasets. Rule of thumb is use all the datasets for the experiments unless you have reason or justification why you excluded certain datasets.

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 TrustWatch, we address the following research questions:

- **RQ1.** How does TrustWatch compare to existing systems in terms of detection performance?
- **RQ2.** How effective is the categorization-based GNN ensemble and Word2vec harmonization scheme for handling data heterogeneity?
- **RQ3.** How does TrustWatch compare to existing FL solutions in addressing data heterogeneity and non-IID challenges?
- **RQ4.** How scalable is TrustWatch in an enterprise-level setting with multiple host machines?
- **RQ5.** How robust is TrustWatch against adversarial mimicry attacks?
- **RQ6.** What is the resource consumption of various components of TrustWatch and its end-to-end processing time on a client machine?

We discuss RQ6 in Appendix C. Additionally, we present a detailed ablation study in Appendix D on the effect of the number of GNN submodels, federated averaging rounds, differential privacy, anomaly threshold, and various system event sizes on our system's scalability.

Implementation. TrustWatch is developed in Python with around 5500 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 [50] 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 developing the semantic attribute encoder, 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 [2], E5 [3], and OpTC [4] 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 2% in E3, and approximately 19% in E5.

The low infiltration rate in OpTC underscores the system's capacity for detecting highly stealthy adversaries, whereas the more widespread attacks in 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, underscoring its suitability for real-world scenarios where logs are sourced from diverse operating systems and exhibit substantial variation in size and complexity.

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 B. ATLASv2 [103] 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 conduct comparisons against existing state-of-the-art PIDS. ThreaTrace [111], a node-level system, employs graph representation learning to identify anomalous nodes within a provenance graph. MAGIC [65] is another recent PIDS which uses masked graph representation learning to detect system threats. Flash [102] is another notable node-level system, which utilizes semantic feature vectors and an

embedding recycling database to offer superior detection performance and efficiency. Flash detection results, as shown in Table 2, have been shown to surpass both ThreaTrace and MAGIC. Hence, our comparison primarily focuses on Flash. Additionally, we include Kairos [33] in our comparison, which utilizes temporal graph networks to capture the evolution of a system's provenance graph over time. We do not compare against Streamspot [88] and Unicorn [52] as they are graph-level detectors, and recent systems like ThreaTrace and Flash have been shown to surpass them in detection performance. While DisDet [38], Prographer [116] and Shadewatcher [121] are notable PIDS, we exclude them from our comparison because DisDet and Prographer are closed-source and a major component of Shadewatcher is proprietary, which hinders our ability to conduct a thorough comparison. It is important to note that, similar to existing works (e.g., Kairos, Shadewatcher, and Prographer), TrustWatch 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 2.

6.1 Detection performance

We conducted experiments to assess how TrustWatch compares with other systems in terms of detection performance. Initially, we outline our methodology for deploying TrustWatch on the DARPA E3, E5, and OpTC datasets. The E3 dataset comprises various scenarios, including Cadets, Theia, and Trace, each representing logs generated by a single host machine. To evaluate TrustWatch on E3, we treat each scenario as an individual host. Consequently, in our federated learning approach, we trained local GNN models on each scenario individually. These local models then participated in federated averaging, a process repeated across 10 rounds. Upon completing the training, we evaluated the global GNN model against the attack logs from these E3 scenarios. Similar to E3, the E5 dataset is also divided into different scenarios; however, each scenario comprises three different hosts. For the OpTC dataset, we used multiple hosts for inclusion in our federated provenance graph learning experiment. Additionally, we also conducted analysis on an enterprise level using this dataset, the results of which are detailed in Section 6.4. For each dataset, we use logs from the benign period to train our system and then evaluate it on the attack logs. The attack logs proceed benign logs in the timeline. For example, the OpTC dataset has six days of benign system logs and three days of attack logs. We run TrustWatch on all days of attack logs to analyze detection performance. For conducting these evaluations, we use the same detection metrics as defined by existing node-level detectors such as ThreaTrace and Flash.

Table 2 reveals that TrustWatch's performance on these datasets is comparable to that of Flash, despite the data heterogeneity, diverse log patterns, and data imbalance contained within each E3, E5, and OpTC datasets. This underscores TrustWatch'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 TrustWatch, poses a challenge for direct comparison. Nevertheless, our results remain competitive with Kairos. Beyond detection performance, we also highlight TrustWatch's qualitative advantages, including its privacy-preserving features and decentralized, scalable operation. These aspects underscore the value of TrustWatch in contrast to centralized systems, emphasizing its high practicality in real-world deployments. Since Kairos was not originally evaluated on the DARPA E3 Trace, we did not compare Kairos on this dataset for fairness because unlike Flash extensive hyperparameter tuning might be needed for Kairos to produce the best results.

Table 3: Effectiveness of categorized GNN 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	0.99	0.96	0.97

6.2 Effectiveness of categorized graph learning & Word2Vec harmonization

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 3 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 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 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 5. Then each client used this centralized model for generating semantic features. Table 4 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 4: Effectiveness of Word2Vec vectors 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	0.99	0.96	0.97

Table 5: Federated averaging algorithms comparison.

Algorithm	Prec.	Rec.	F-Score
TrustWatch	0.90	0.92	0.91
FedAvg	0.58	0.97	0.73
FedProx	0.64	0.95	0.76
FedOpt	0.63	0.90	0.74

6.3 Comparison with existing FL solutions for heterogeneity.

Wajih: Use all the datasets for this experiment. We can't cherry pick certain datasets. It is a very easy way to kill the paper.

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 [77] and FedOpt [22]. 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 TrustWatch variant of FedAvg, which incorporates semantic harmonization and categorized learning. Table 5 presents the evaluation results. Both FedProx and FedOpt outperform FedAvg; however, neither matches the performance of TrustWatch.

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.

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.

6.4 Scalability in Enterprise settings

We analyzed the performance of our system in an enterprise setting using the OpTC dataset, which includes a large number of host machines. We compared our system to centralized systems such as Flash and Kairos, focusing on important metrics such as network and processing overhead.

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, TrustWatch achieves a significant reduction in these overheads. The only network expenses for TrustWatch 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 TrustWatch is minimal compared to centralized systems where the raw system logs need to be sent over the network. Ultimately, TrustWatch 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, TrustWatch 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.

7 ADVERSARIAL ATTACKS ANALYSIS

In this section, we analyze the robustness of our system against mimicry, model poisoning, and gradient-based attacks. Membership inference attacks are discussed in Section 8.

Wajih: explain why we didn't use the other adversarial attack https://www.usenix.org/conference/usenixsecurity23/presentation/mukherjee

Mimicry Attacks. We evaluated our system's resilience against the adversarial mimicry attack detailed by Goyal et al. [47] and compared its robustness with Flash. The attack aims to mimic benign graph embeddings, integrating benign node structures to evade detection. Using the E3 dataset, similar to Flash for fair comparison, our findings (shown in Figure 4) 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. Conversely, Flash, relying on a generalized model for anomaly detection, exhibits vulnerabilities to mimicry attacks, as it may miss critical details. As demonstrated by the authors, Flash initially experiences a drop in anomaly scores, increasing the likelihood of attack nodes evading detection—a vulnerability not present in our system.

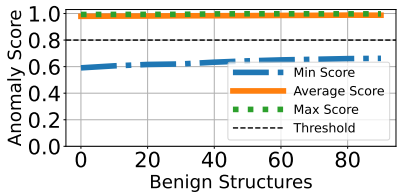
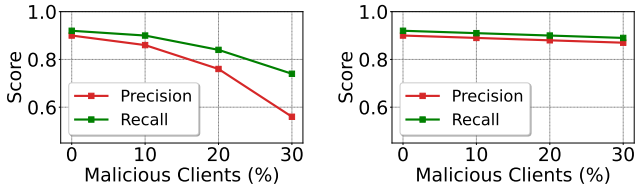


Figure 4: Adversarial mimicry attack analysis.



(a) FedAvg aggregation. **(b) Multi-Krum aggregation**

Figure 5: Model poisoning attack analysis.

Model Poisoning Attacks. These occur when one or more malicious clients submit corrupted local model updates to 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 [94]—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 [31] 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 [107] is one effective strategy in which the model is trained on perturbed input data, thereby increasing its robustness against these attacks.

8 PRIVACY PRESERVATION ANALYSIS

In this section, we analyze the preservation of user privacy within TrustWatch, which is structured around three primary components: the central server, the utility server, and clients. The risk to TrustWatch's privacy arises from the possibility of inference attacks using model weights at the central server and Word2Vec tokens at the utility server. These attacks aim to infer whether some system entity or attributes were used in the training data or not. We will discuss the scope and limitations of these attacks on TrustWatch below:

Component Roles. The central server's role involves the application of federated averaging to the GNN models received from clients. The utility server performs contextual aggregation of semantic attribute vectors derived from clients' $Word2Vec$ models. The mathematical representation for averaging vectors of a token k across N clients is given by: $\bar{v}_k = \frac{1}{N}\sum_{i=1}^{N} v_{k,i}$ where $\bar{v}_k$ is the averaged vector for token k, and $v_{k,i}$ is the vector representation of token k from the i-th client model. Clients are tasked with training the Word2Vec and GNN models on provenance graphs, these graphs are constructed from their local system audit logs.

Privacy Risk Analysis. Within TrustWatch, potential privacy compromises arise if either the central or utility server can infer specific details about individual clients' logs, such as the applications in use or particular attributes like filenames and IP addresses. Despite the central server's inability to access raw client data directly, it receives model updates from clients, thereby introducing a vulnerability to model inference attacks through analysis of these updates.

Consider an attacker's objective to ascertain whether a system entity x with attributes y was utilized in training a client model m. This necessitates the generation of multiple candidate node features for x, taking into account various graph structures and interactions with other entities while considering diverse attributes. The search space for this task, S, is extensive, spanning all conceivable processes, files, and network IPs.

Assuming the server generates multiple candidate structures, it then requires access to the specific client's `Word2Vec` model to generate feature vectors for these structures—a step prevented by the model's unavailability and inherent algorithmic randomness, rendering each training iteration of the `Word2Vec` model distinct:

$$F(x) = \texttt{Word2Vec}(s_x)$$

where $F(x)$ is the feature vector of structure x and s_x is the candidate sequence.

Formally, the system TrustWatch is designed to ensure that the probability that an adversary can distinguish between two arbitrary system log datasets based on the observed updates is negligible:

$$\begin{aligned}\text{Attack}(D_1, D_2) = &|\Pr[\mathcal{A}(\text{Enc}(D_1)) = 1] \\ &- \Pr[\mathcal{A}(\text{Enc}(D_2)) = 1]| \\ \leq\ &\epsilon\end{aligned}$$

where ϵ is a negligible value, $\text{Enc}(D)$ denotes the encoding of the dataset into model updates, and $\mathcal{A}$ represents the adversary.

To further quantify the security of TrustWatch in the context of semantic embeddings at the utility server, consider the scenario where an attacker has access to the Word2Vec embeddings but

not the original attributes used for generating them. Suppose the attacker attempts to reverse engineer the embedding vectors to recover the original system log data. The complexity, randomness and high dimensionality of the embedding space, combined with the non-linear transformations typically applied during Word2Vec embedding process, ensure that the probability of successfully identifying the original tokens from the embeddings is negligibly small:

$$\Pr[\text{Rev}(\mathcal{E}, e) = t] \leq \frac{1}{|T|}$$

where Rev denotes the hypothetical reverse mapping function from embeddings to tokens, $\mathcal{E}$ represents the embedding process, e is the observed embedding, t is the original token, and $|T|$ is the cardinality of the token set. This probability indicates that correctly guessing the original token from its embedding is as likely as randomly selecting one token out of the entire set of possible tokens, which is practically infeasible given a sufficiently large and diverse token set.

Consequently, the dual-server architecture and semantic featurization significantly limit the central server's capacity for inference attacks. The utility server's function of contextually aggregating semantic attribute vectors introduces another potential privacy concern if the server could deduce the entities these vectors represent. This risk is mitigated by the secure encryption of tokens using keys generated by the central server, which the utility server cannot access, ensuring privacy protection.

Furthermore, a malicious client attempting to utilize the global aggregate to infer the applications used across the organization faces similar constraints. Such an attack is limited by the client's lack of access to the semantic attribute vectors widespread across the organization, significantly narrowing the attack's feasibility.

Therefore, the architecture of TrustWatch robustly defends against model inference attacks, affirming the system's capacity to preserve privacy. This resilience is predicated on the non-collusion assumption between the central and utility servers—a standard premise upheld by related works [104, 113], ensuring the system's high degree of privacy preservation.

9 DISCUSSION & LIMITATIONS

Alert Investigation. In systems like TrustWatch, validating the veracity of alerts is crucial for ensuring system reliability and preventing alert fatigue [55], traditionally involving security analysts manually reviewing each alert based on activities within local provenance graphs. This manual process, while thorough, is time-consuming and prone to errors, posing challenges in scalability and privacy. To enhance privacy and maintain efficiency in alert verification, implementing privacy-preserving techniques such as Secure Multi-party Computation (SMC) [45], Homomorphic Encryption (HE) [119], and Zero-Knowledge Proofs (ZKP) [42] is vital. SMC allows multiple stakeholders to collaboratively validate alerts by jointly computing functions over their inputs while keeping those inputs private. HE enables the central server to analyze encrypted data, ensuring data confidentiality, and ZKP offers a way for one party to prove the validity of an alert to another without revealing any other information. Leveraging these technologies allows TrustWatch to privately and securely validate alerts, enhancing the system's trustworthiness and optimizing operational workflow. 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 security analysts to perform more in-depth attack analysis.

Explainability. The use of deep learning models in PIDS makes them black-box systems, making it difficult to explain the inner decision-making process of these systems. Similar to existing PIDS [33, 102, 116], TrustWatch also suffers from this interpretability problem. This issue slows the adoption of these PIDS compared to rule-based systems. Existing techniques [18, 20, 28, 60] for model explainability focus on metrics such as feature importance. However, modern PIDS use complex feature engineering techniques. For example, Flash uses a Word2Vec model for feature generation from textual attributes in audit logs. These features are then used in a Graph Neural Network model for anomaly detection. The use of this multi-model approach renders existing explainable AI techniques impractical for these PIDS. We identify explainable PIDS as an important area for future research. The recent advancements in Large Language Models (LLMs) [32] from the domain of Natural Language Processing can be utilized to solve this challenge. Future work can focus on using the reasoning power of LLMs to develop techniques for explaining PIDS outputs.

Log Retention. Effective log retention [112] is critical for the performance and reliability of PIDS. Advanced Persistent Threats (APTs) can span months; therefore, logs need to be stored for extended periods to allow for thorough attack investigations. Decentralized systems like TrustWatch and DisDet [38] keep raw logs on client machines. However, client machines have limited storage, so they can only store logs for a limited period. The duration for storing these logs depends on the domain in which TrustWatch is deployed. Secure and tamper-resistant cloud storage solutions [69] can be combined with TrustWatch to securely back up user logs. In the event of an attack investigation, the logs corresponding to the alert raised by TrustWatch can be fetched from storage. These logs can then be processed to remove sensitive attributes to preserve user privacy [99]. These logs can then be used for attack investigation.

Concept Drift. This is a problem where the data distribution of the underlying system evolves over time. For instance, with the emergence of new system activities, the patterns learned by TrustWatch during training might not remain valid. This drift could lead to misclassifications, as new benign behaviors might be mistakenly identified as anomalies. One mitigation strategy for this involves periodic retraining with more recent data to update the models. Techniques mentioned in recent works [24, 66, 80] can be leveraged to deal with the problem of concept drift.

Unseen Tokens in Semantic Encoder. In TrustWatch, we rely on Word2Vec models to encode semantic attributes. Because Word2Vec operates inductively, it only provides embeddings for tokens it has previously learned, which can diminish the quality of feature vectors for new nodes whose attributes contain unseen tokens. To represent each node, we construct a word sequence by combining its semantic attributes with the system calls observed in neighboring nodes. However, for new nodes with previously unseen attribute tokens, the sequence is largely limited to system

calls—similar to related systems such as ThreaTrace [111]. To address this challenge, we can retrain the Word2Vec model more frequently or adopt subword-level embedding models, such as fastText [67], which split words into subword units to generate embeddings for previously unseen tokens. Additionally, tokenization methods like Byte-Pair Encoding [19] can further break down unknown words into smaller subword components, thereby improving new token coverage.

10 CONCLUSION

In this paper, we introduced TrustWatch, a privacy-preserving system for intrusion detection that leverages secure model harmonization and a personalized provenance graph learning framework to tackle data heterogeneity and performance challenges in federated learning for PIDS settings. Our evaluations on open-source datasets show that TrustWatch matches the performance of centrally trained PIDS while preserving user privacy and providing better scalability.

REFERENCES

[1] [n. d.]. Audit Logging Overview. https://www.datadoghq.com/knowledge-center/audit-logging/.
[2] [n. d.]. DARPA Engagement 3. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.
[3] [n. d.]. DARPA Engagement 5. https://github.com/darpa-i2o/Transparent-Computing/tree/master.
[4] [n. d.]. DARPA OPTC. https://github.com/FiveDirections/OpTC-data.
[5] [n. d.]. DTrace. https://www.freebsd.org/doc/handbook/dtrace.html.
[6] [n. d.]. Event tracing. https://docs.microsoft.com/en-us/windows/desktop/ETW/event-tracing-portal.
[7] [n. d.]. IT security economics. https://www.kaspersky.com/blog/it-security-economics-2020-part-4/.
[8] [n. d.]. The Linux audit daemon. https://linux.die.net/man/8/auditd.
[9] [n. d.]. NotPetya Attack. https://en.wikipedia.org/wiki/Petya_and_NotPetya.
[10] [n. d.]. The SolarWinds Cyber-Attack: What You Need to Know. https://www.cisecurity.org/solarwinds/.
[11] [n. d.]. What is an Instance in Cloud Computing? https://scaleyourapp.com/what-is-an-instance-in-cloud-computing-a-thorough-guide/.
[12] Adil Ahmad, Sangho Lee, and Marcus Peinado. 2022. HARDLOG: Practical Tamper-Proof System Auditing Using a Novel Audit Device. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society.
[13] Shadi Aljawarneh, Monther Aldwairi, and Muneer Bani Yassein. 2018. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science* 25 (2018).
[14] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. 2021. ATLAS: A Sequence-based Learning Approach for Attack Investigation. In *USENIX Security Symposium*.
[15] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, Abhi Shelat, and Ivan Visconti. 2009. 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, 524–540.
[16] Md Monowar Anjum, Shahrear Iqbal, and Benoit Hamelin. 2021. 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*.
[17] Meenatchi Sundaram Muthu Selva Annamalai, Igor Bilogrevic, and Emiliano De Cristofaro. 2023. FP-Fed: Privacy-Preserving Federated Detection of Browser Fingerprinting. *arXiv preprint arXiv:2311.16940* (2023).
[18] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. 2021. Explaining anomalies detected by autoencoders using Shapley Additive Explanations. *Expert systems with applications* 186 (2021), 115736.
[19] Ali Araabi, Christof Monz, and Vlad Niculae. 2022. How effective is byte pair encoding for out-of-vocabulary words in neural machine translation? *arXiv preprint arXiv:2208.05225* (2022).
[20] Carmelo Ardito, Yashar Deldjoo, Eugenio Di Sciascio, and Fatemeh Nazary. 2021. Revisiting Security Threat on Smart Grids: Accurate and Interpretable Fault Location Prediction and Type Classification.. In *ITASEC*. 523–533.
[21] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A Reuter, and Martin Strand. 2015. A guide to fully homomorphic encryption. *Cryptology ePrint Archive* (2015).
[22] Muhammad Asad, Ahmed Moustafa, and Takayuki Ito. 2020. Fedopt: Towards communication efficiency and privacy preservation in federated learning. *Applied Sciences* 10, 8 (2020), 2864.
[23] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2013. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 535–548.
[24] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. 2022. Transcending transcend: Revisiting malware classification in the presence of concept drift. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 805–823.
[25] Adam Bates, Wajih Ul Hassan, Kevin Butler, Alin Dobra, Bradley Reaves, Patrick Cable, Thomas Moyer, and Nabil Schear. 2017. Transparent web service auditing via network provenance functions. In *International World Wide Web Conference (WWW)*.
[26] Adam Bates, Dave Jing Tian, Kevin RB Butler, and Thomas Moyer. 2015. Trustworthy whole-system provenance for the linux kernel. In *USENIX Security Symposium*.
[27] Mohammad Ubaidullah Bokhari and Qahtan Makki Shallal. 2016. A review on symmetric key encryption techniques in cryptography. *International journal of computer applications* 147, 10 (2016).
[28] Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. 2018. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the first workshop on machine learning for computing systems*. 1–8.
[29] Riadh Ben Chaabene, Darine Ameyed, Fehmi Jaafer, Alexis Roger, Aimeur Esma, and Mohamed Cheriet. 2023. A privacy-preserving federated learning for IoT intrusion detection system. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 351–356.
[30] S Sibi Chakkaravarthy, D Sangeetha, and V Vaidehi. 2019. A survey on malware analysis and mitigation techniques. *Computer Science Review* 32 (2019).
[31] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology* 6, 1 (2021), 25–45.
[32] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.
[33] Zijun Cheng, Qiujian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. 2023. Kairos:: Practical Intrusion Detection and Investigation using Whole-system Provenance. *arXiv preprint arXiv:2308.05034* (2023).
[34] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. 2020. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review* 53 (2020), 5113–5155.
[35] KR1442 Chowdhary and KR Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.
[36] Ronald Cramer, Ivan Bjerre Damgård, et al. 2015. *Secure multiparty computation*. Cambridge University Press.
[37] Vishnu Asutosh Dasu, Sumanta Sarkar, and Kalikinkar Mandal. 2022. PROV-FL: Privacy-preserving round optimal verifiable federated learning. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*. 33–44.
[38] Feng Dong, Liu Wang, Xu Nie, Fei Shao, Haoyu Wang, Ding Li, Xiapu Luo, and Xusheng Xiao. 2023. *DISTDET*: A *Cost-Effective* Distributed Cyber Threat Detection System. In *32nd USENIX Security Symposium (USENIX Security 23)*. 6575–6592.
[39] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs Through Deep Learning. In *ACM Conference on Computer and Communications Security (CCS)*.
[40] Wenliang Du and Mikhail J Atallah. 2001. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*. 13–22.
[41] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. 2020. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 59–71.
[42] Uriel Fiege, Amos Fiat, and Adi Shamir. 1987. Zero knowledge proofs of identity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 210–217.
[43] Othmane Friha, Mohamed Amine Ferrag, Mohamed Benbouzid, Tarek Berghout, Burak Kantarci, and Kim-Kwang Raymond Choo. 2023. 2DF-IDS: Decentralized and differentially private federated learning-based intrusion detection system for industrial IoT. *Computers & Security* 127 (2023), 103097.
[44] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 169–178.

[45] Oded Goldreich. 1998. Secure multi-party computation. *Manuscript. Preliminary version* 78, 110 (1998), 1–108.

[46] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[47] Akul Goyal, Xueyuan Han, Gang Wang, and Adam Bates. 2023. Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems. In *30th Network and Distributed System Security Symposium*.

[48] Wei Guo, Zhiwei Yao, Yongfei Liu, Lanxue Zhang, Liangxiong Li, Tong Li, and Bingzhen Wu. 2023. 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, 494–500.

[49] Manasi Gyanchandani, JL Rana, and RN Yadav. 2012. Taxonomy of anomaly based intrusion detection system: a review. *International Journal of Scientific and Research Publications* 2 (2012).

[50] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[51] Dongqi Han, Zhiliang Wang, Wenqi Chen, Ying Zhong, Su Wang, Han Zhang, Jiahai Yang, Xingang Shi, and Xia Yin. 2021. 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*.

[52] Xueyan Han, Thomas Pasqueir, Adam Bates, James Mickens, and Margo Seltzer. 2020. Unicorn: Runtime Provenance-Based Detector for Advanced Persistent Threats. In *Network and Distributed System Security (NDSS)*.

[53] Xueyuan Han, Xiao Yu, Thomas Pasquier, Ding Li, Junghwan Rhee, James Mickens, Margo Seltzer, and Haifeng Chen. 2021. SIGL: Securing Software Installations Through Deep Graph Learning. In *USENIX Security Symposium*.

[54] Wajih Ul Hassan, Adam Bates, and Daniel Marino. 2020. Tactical Provenance Analysis for Endpoint Detection and Response Systems. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE.

[55] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. NoDoze: Combatting threat alert fatigue with automated provenance triage. In *Network and Distributed System Security (NDSS)*.

[56] Wajih Ul Hassan, Mark Lemay, Nuraini Aguse, Adam Bates, and Thomas Moyer. 2018. Towards scalable cluster auditing through grammatical inference over provenance graphs. In *Network and Distributed System Security (NDSS)* (San Diego, CA).

[57] Wajih Ul Hassan, Mohammad Ali Noureddine, Pubali Datta, and Adam Bates. 2020. OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis. In *NDSS*.

[58] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott D Stoller, and VN Venkatakrishnan. 2017. SLEUTH: Real-time attack scenario reconstruction from COTS audit data. In *USENIX Security Symposium*.

[59] Md Nahid Hossain, Junao Wang, R. Sekar, and Scott D. Stoller. 2018. Dependence-Preserving data compaction for scalable forensic analysis. In *USENIX Security Symposium*.

[60] Chanwoong Hwang and Taejin Lee. 2021. E-SFD: Explainable sensor fault detection in the ICS anomaly detection system. *IEEE Access* 9 (2021), 140470–140486.

[61] Muhammad Adil Inam, Yinfang Chen, Akul Goyal, Jason Liu, Jaron Mink, Noor Michael, Sneha Gaur, Adam Bates, and Wajih Ul Hassan. 2023. Sok: History is a vast early warning system: Auditing the provenance of system intrusions. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2620–2638.

[62] Muhammad Adil Inam, Wajih Ul Hassan, Ali Ahad, Adam Bates, Rashid Tahir, Tianyin Xu, and Fareed Zaffar. 2022. Forensic Analysis of Configuration-based Attacks. In *NDSS*.

[63] EL GAABOURI Ismail, Asaad Chahboun, and Naoufal Raissouni. 2020. Fernet Symmetric Encryption Method to Gather MQTT E2E Secure Communications for IoT Devices. (2020).

[64] Takamasa Isohara, Keisuke Takemori, and Ayumu Kubota. 2011. Kernel-based behavior analysis for android malware detection. In *2011 seventh international conference on computational intelligence and security*. IEEE.

[65] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. 2023. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. *arXiv preprint arXiv:2310.09831* (2023).

[66] Roberto Jordaney, Kumar Sharad, Santanu K Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. 2017. Transcend: Detecting concept drift in malware classification models. In *26th USENIX security symposium (USENIX security 17)*. 625–642.

[67] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

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

[69] Manish Kumar, Ashish Kumar Singh, and TV Suresh Kumar. 2018. Secure log storage using blockchain and cloud infrastructure. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 1–4.

[70] Yonghwi Kwon, Fei Wang, Weihang Wang, Kyu Hyung Lee, Wen-Chuan Lee, Shiqing Ma, Xiangyu Zhang, Dongyan Xu, Somesh Jha, Gabriela Ciocarlie, et al. 2018. MCI: Modeling-based Causality Inference in Audit Logging for Attack Investigation. In *Network and Distributed System Security (NDSS)*.

[71] Marie Laclau, Ludovic Renou, and Xavier Venel. 2020. Robust communication on networks. *arXiv preprint arXiv:2007.00457* (2020).

[72] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[73] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. PMLR.

[74] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2013. High accuracy attack provenance via binary-based execution partition. In *Network and Distributed System Security (NDSS)* (San Diego, CA).

[75] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2013. LogGC: garbage collecting audit log. In *CCS*. 12 pages.

[76] Jianbin Li, Xin Tong, Jinwei Liu, and Long Cheng. 2023. An Efficient Federated Learning System for Network Intrusion Detection. *IEEE Systems Journal* (2023).

[77] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[78] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. 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*. 1777–1794.

[79] Yushan Liu, Mu Zhang, Ding Li, Kangkook Jee, Zhichun Li, Zhenyu Wu, Junghwan Rhee, and Prateek Mittal. 2018. Towards a Timely Causality Analysis for Enterprise Security. In *Network and Distributed System Security (NDSS)*.

[80] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering* 31, 12 (2018), 2346–2363.

[81] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133* (2020).

[82] Shiqing Ma, Kyu Hyung Lee, Chung Hwan Kim, Junghwan Rhee, Xiangyu Zhang, and Dongyan Xu. 2015. Accurate, low cost and instrumentation-free security audit logging for Windows. In *Annual Computer Security Applications Conference (ACSAC)*.

[83] Shiqing Ma, Juan Zhai, Yonghwi Kwon, Kyu Hyung Lee, Xiangyu Zhang, Gabriela Ciocarlie, Ashish Gehani, Vinod Yegneswaran, Dongyan Xu, and Somesh Jha. 2018. Kernel-supported cost-effective audit logging for causality tracking. In *USENIX Annual Technical Conference (ATC)*.

[84] Shiqing Ma, Juan Zhai, Fei Wang, Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2017. MPI: Multiple perspective attack investigation with semantic aware execution partitioning. In *USENIX Security Symposium*.

[85] Shiqing Ma, Juan Zhai, Fei Wang, Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2017. MPI: Multiple Perspective Attack Investigation with Semantic Aware Execution Partitioning. In *USENIX Security Symposium*.

[86] Shiqing Ma, Xiangyu Zhang, and Dongyan Xu. 2016. ProTracer: Towards practical provenance tracing by alternating between logging and tainting. In *Network and Distributed System Security (NDSS)* (San Diego, CA).

[87] Dapeng Man, Fanyi Zeng, Wu Yang, Miao Yu, Jiguang Lv, and Yijing Wang. 2021. Intelligent intrusion detection based on federated learning for edge-assisted internet of things. *Security and Communication Networks* 2021 (2021), 1–11.

[88] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[89] Ziadoon Kamil Maseer, Robiah Yusof, Nazrulazhar Bahaman, Salama A Mostafa, and Cik Feresa Mohd Foozy. 2021. Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset. *IEEE access* 9 (2021).

[90] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. 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*. 1–9.

[91] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. 2018. *Handbook of applied cryptography*. CRC press.

[92] Sadegh M. Milajerdi, Birhanu Eshete, Rigel Gjomemo, and V.N. Venkatakrishnan. 2019. POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting. In *ACM Conference on Computer and Communications Security (CCS)*.

[93] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan. 2019. HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows. In *IEEE Symposium on Security and Privacy (S&P)*.

[94] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. 2019. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125* (2019).
[95] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 113–124.
[96] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).
[97] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. 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, 739–753.
[98] Riccardo Paccagnella, Pubali Datta, Wajih Ul Hassan, Adam Bates, Christopher Fletcher, Andrew Miller, and Dave Tian. 2020. Custos: Practical tamper-evident auditing of operating systems using trusted execution. In *Network and distributed system security symposium*.
[99] A Omar Portillo-Dominguez and Vanessa Ayala-Rivera. 2019. 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, 107–115.
[100] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. 2022. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10061–10071.
[101] Vinay Raj, Satharla Suresh, and MSB Phridviraj. 2023. Performance Analysis of Hybrid Cryptographic Algorithms in Serverless Platforms. In *International Conference on Advanced Computing and Intelligent Technologies*. Springer, 93–105.
[102] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. 2024. FLASH: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning. In *IEEE Symposium on Security and Privacy (S&P)*.
[103] Andy Riddle, Kim Westfall, and Adam Bates. 2023. ATLASv2: ATLAS Attack Engagements, Version 2. arXiv:2401.01341 [cs.CR]
[104] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. 2020. Crypte: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 603–619.
[105] Sinem Sav, Apostolos Pyrgelis, Juan R Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. 2020. POSEIDON: Privacy-preserving federated neural network learning. *arXiv preprint arXiv:2009.00349* (2020).
[106] Yun Shen and Gianluca Stringhini. 2019. Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *USENIX Security Symposium*.
[107] Florian Tramer and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems* 32 (2019).
[108] Baocang Wang, Yange Chen, Hang Jiang, and Zhen Zhao. 2023. Ppefl: Privacy-preserving edge federated learning with local differential privacy. *IEEE Internet of Things Journal* (2023).
[109] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. 2018. Fear and Logging in the Internet of Things. In *Network and Distributed System Security (NDSS)*.
[110] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. 2020. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis.. In *Network and Distributed System Security (NDSS)*.
[111] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. 2022. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security* 17 (2022).
[112] Brittany Wilbert and Lei Chen. 2012. 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 …, 1.
[113] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications* 13, 1 (2022), 3091.
[114] Bin Xia, Junjie Yin, Jian Xu, and Yun Li. 2019. 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, 61–76.
[115] Wenjun Xiong, Emeline Legrand, Oscar Åberg, and Robert Lagerström. 2022. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. *Software and Systems Modeling* 21, 1 (2022), 157–177.
[116] Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. 2023. PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding. (2023).
[117] Yuchen Yang, Bo Hui, Haolin Yuan, Neil Gong, and Yinzhi Cao. 2023. {PrivateFL}: Accurate, differentially private federated learning via personalized data transformation. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1595–1612.
[118] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. 2020. Federated learning in vehicular edge computing: A selective model aggregation approach. *IEEE Access* 8 (2020), 23920–23935.
[119] Xun Yi, Russell Paulet, Elisa Bertino, Xun Yi, Russell Paulet, and Elisa Bertino. 2014. *Homomorphic encryption*. Springer.
[120] Oualid Zari, Chuan Xu, and Giovanni Neglia. 2021. Efficient passive membership inference attack in federated learning. *arXiv preprint arXiv:2111.00430* (2021).
[121] Jun Zeng, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. 2022. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *IEEE Symposium on Security and Privacy (SP)*.
[122] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).
[123] Tailin Zhou, Jun Zhang, and Danny HK Tsang. 2023. FedFA: Federated Learning with Feature Anchors to Align Features and Classifiers for Heterogeneous Data. *IEEE Transactions on Mobile Computing* (2023).
[124] Mohamad Fadli Zolkipli and Aman Jantan. 2011. An approach for malware behavior identification and classification. In *2011 3rd international conference on computer research and development*, Vol. 1. IEEE.

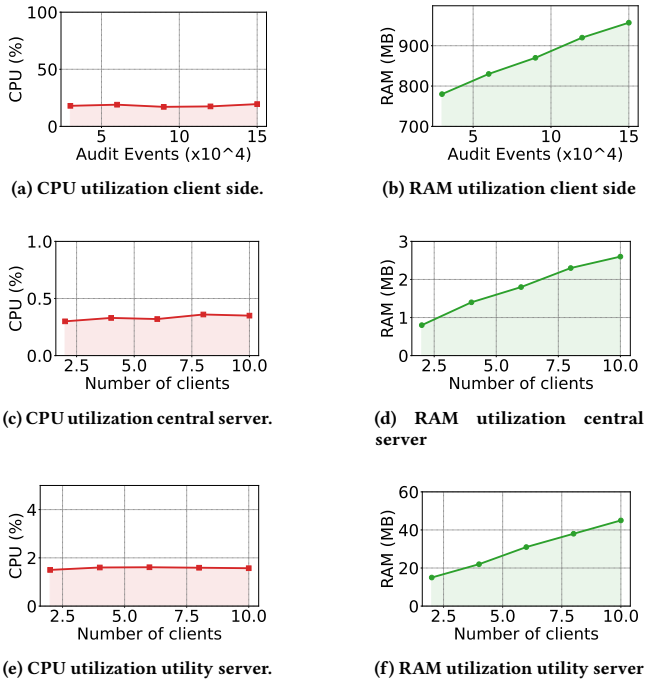
A PROVENANCE GRAPH CONSTRUCTOR

TrustWatch utilizes system logs to construct a system provenance graph. It operates on each client machine, using their local system logs to build the graph. Major operating systems, including Linux and Windows, come equipped with built-in mechanisms for log collection – specifically, the Linux Audit system [8] and Windows Event Tracing [6]. These logs provide detailed insights into the interactions among various system entities, capturing activities, such as process executions, file operations, and network connections. Using this data, TrustWatch forms a graph where nodes represent system entities including processes, files, and sockets. The edges of this graph denote events, identified by syscalls, that occur between these entities. Moreover, TrustWatch enhances each node with comprehensive attributes, including process names, command lines, file names, and network IP addresses. As demonstrated by prior works [33, 102], these contextual attributes enable the model to develop a robust understanding of system behavior.

B DATASET DESCRIPTION

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.

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 [115] 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



(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 TrustWatch.

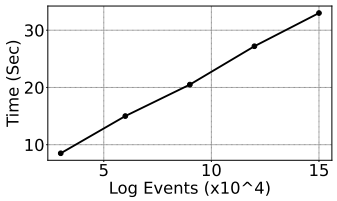


Figure 7: Processing time for various audit event sizes evaluated using OpTC dataset.

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.

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.

C 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 TrustWatch. 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 TrustWatch's resource consumption is moderate. Specifically, TrustWatch 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 TrustWatch 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 with TrustWatch to measure the time taken to process these events on a client machine. The results, illustrated in Figure 7, demonstrate that TrustWatch 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, TrustWatch 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.

D ABLATION STUDY

In this ablation study, we analyze the impact of key parameters within TrustWatch. 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:

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.

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 personalized GNN learning framework, k controls the creation of distinct standardized bins. These bins categorize processes across client

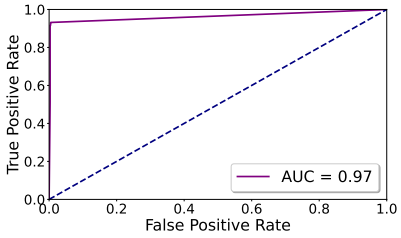


Figure 10: Anomaly threshold effect.

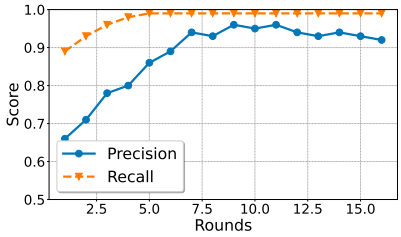


Figure 8: Federated averaging rounds vs detection performance using E3 dataset.

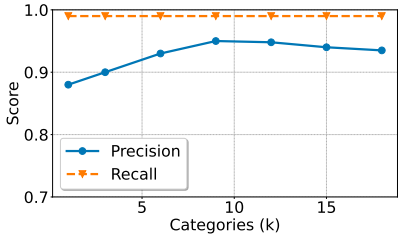


Figure 9: Effect of number of categories vs detection performance using E3 dataset.

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.

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 D, 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.

Anomaly Threshold

This hyperparameter controls the number of alerts generated by TrustWatch. It determines the trade-off between precision and recall. We use the E5 dataset to investigate the effects of different anomaly thresholds. Figure 10 illustrates the results.

Effect of differential privacy on accuracy

In Section 8, 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 [81, 97, 120], 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 TrustWatch. 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 5.

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 TrustWatch and instead design a dual-server architecture to offer privacy guarantees without adding noise to the model updates.

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

Algorithm 2: Federated Provenance Graph Learning

Inputs : Set of client models $\{GNN_1, GNN_2, \ldots, GNN_N\}$;
Output : Global GNN model GNN_{Global}.

/* Initialize global GNN model with random weights for a given process Category. */
$G \leftarrow$ InitializeRandomWeights()
/* Distribute GNN to all clients. */
foreach *client* **do**
    Send GNN to $Client_i$
    $GNN \leftarrow$ TrainOnLocalData($ProcessCategory_i$)
end
/* Aggregate trained models from clients. */
$AggregatedWeights \leftarrow list([])$
foreach *client model* GNN_i **do**
    $AggregatedWeights.append$(ExtractWeights(GNN_i))
end
/* Apply federated averaging. */
$GNN_{Global} \leftarrow$ FederatedAveraging($AggregatedWeights$)
return GNN_{Global}

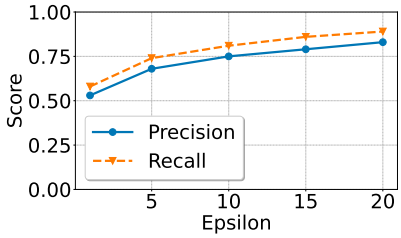


Figure 11: Effect of differential privacy noise on detection using E3 dataset. Note that we observed similar results on the other datasets.

with an HMAC for authenticity. Because the HMAC relies on efficient hash functions rather than expensive asymmetric operations [101], 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 [91]. Their computational overhead makes them impractical for scenarios involving substantial data, such as Word2Vec models with thousands of tokens.

This efficiency is crucial in TrustWatch, 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 [95]. 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.

E 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. TrustWatch 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 [34], selective update aggregation [118] and the implementation of more robust communication protocols [71] can significantly improve handling large volumes of updates, thus preventing state explosion and maintaining system performance in large-scale federated learning environments.