ENTENTE: Cross-silo Intrusion Detection on Network Log Graphs with Federated Learning

Jiacen Xu*[§], Chenang Li[†], Yu Zheng[†] and Zhou Li[†]
[*] Microsoft, Email: jiacenxu@microsoft.com
[†] University of California, Irvine, Email: {chenangl, yu.zheng, zhou.li}@uci.edu

***Abstract*—Graph-based Network Intrusion Detection Systems (GNIDS) have gained significant momentum in detecting sophisticated cyber-attacks, such as Advanced Persistent Threats (APTs), within and across organizational boundaries. Though achieving satisfying detection accuracy and demonstrating adaptability to ever-changing attacks and normal patterns, existing GNIDS predominantly assume a centralized data setting. However, flexible data collection is not always realistic or achievable due to increasing constraints from privacy regulations and operational limitations.**

We argue that the practical development of GNIDS requires accounting for distributed collection settings and we leverage Federated Learning (FL) as a viable paradigm to address this prominent challenge. We observe that naively applying FL to GNIDS is unlikely to be effective, due to issues like graph heterogeneity over clients and the diverse design choices taken by different GNIDS. We address these issues with a set of novel techniques tailored to graph datasets, including reference graph synthesis, graph sketching and adaptive contribution scaling, eventually developing a new system ENTENTE. By leveraging the domain knowledge, ENTENTE can achieve effectiveness, scalability and robustness simultaneously. Empirical evaluation on the large-scale LANL, OpTC and Pivoting datasets shows that ENTENTE outperforms the SOTA FL baselines. We also evaluate ENTENTE under FL poisoning attacks tailored to the GNIDS setting, showing the robustness by bounding the attack success rate to low values. Overall, our study suggests a promising direction for building cross-silo GNIDS.

I. INTRODUCTION

The techniques and scale of modern cyber-attacks are evolving at a rapid pace. More high-profile security breaches are observed against large organizations nowadays. One prominent attack strategy is Advanced Persistent Threat (APT) [1], which establishes multiple attack stages and infiltrates multiple organizational assets through techniques like lateral movement. As a popular countermeasure, many organizations collect network logs (e.g., firewall and proxy logs) and perform intrusion detection on them [2]. To more precisely capture the distinctive network communication patterns of the attack, a promising approach is to model the network logs as a graph and apply graph-based algorithms to detect abnormal entities, interactions or communities. We term such system *Graph-based Network Intrusion Detection System (GNIDS)*, and we observe that the recent works [3], [4], [5], [6] prefer advanced graph models like graph autoencoder (GAE) [7] to build their systems, showing much higher detection accuracy over the traditional NIDS and capabilities of detecting sophisticated attacks like lateral movement [5].

[§]This work was done when the author was a PhD student at UC Irvine.

Regulation compliance concerns for GNIDS. Given the sensitive nature of network logs, such as revealing communication patterns of employees and organizations [8], privacy regulations have to be followed when training GNIDS models with logs from multiple regions. For example, Menges et al. state that SIEM needs to be compliant with Europe's General Data Protection Regulation (GDPR) [9], which covers the data processing and transfer "within and between private companies and/or public bodies in the European member states". Outside of Europe, Singapore's Personal Data Protection Act (PDPA) prohibits using data for purposes beyond its original intent without explicit individual consent [10], creating barriers for training models on network logs.

In fact, the data collection capabilities of a cyber-security company have already been restricted under data privacy regulations. For instance, Palo Alto Networks (PANW) offers a Strata Logging Service that enables enterprises to send on-premise firewall logs to the cloud for centralized analysis and management. However, PANW explicitly states that if regulations mandate data residency, customers must ensure that logs are stored in region-appropriate cloud instances to comply with jurisdictional boundaries [11]. In such cases, logs cannot be aggregated across regions, making centralized analysis on logs impractical. Similarly, Microsoft's Windows Defender XDR stores customer data, such as alerts, in regional Microsoft Azure data centers (e.g., EU, UK, US), and its documentation confirms that cloud tenants cannot be relocated across regions once created [12]. Our communication with product representatives verified that cross-region data centralization is unsupported.

Federated Learning for GNIDS and ENTENTE. The aforementioned compliance issue calls for a new paradigm that allows the development of GNIDS over geographically distributed logs while aligning with various privacy regulations. One promising solution is Federated Learning (FL), which has gained prominent attention from academia and industry [13]. In essence, FL allows the individual data owners (e.g., a device

Network and Distributed System Security (NDSS) Symposium 2026
23 - 27 February 2026 , San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230093
www.ndss-symposium.org

owner or an organization) to keep their data on premise and jointly train a global model by exchanging parameters of local models. Given its successes in addressing privacy concerns of data collection [13], we pivot the research of developing a practical FL-empowered GNIDS and term our new system ENTENTE.

We argue that ENTENTE should satisfy three main design goals: *effectiveness* (similar effectiveness as the GNIDS trained on the entire dataset), *scalability* (the overhead introduced by the FL mechanism should be small and a large number of FL clients should be supported) and *robustness* (maintaining detection accuracy against attackers who compromise the FL procedure). Since the data to be processed by GNIDS usually have imbalanced classes (e.g., malicious events are far less than the normal events) and non-IID (not independent and identically distributed) across FL clients, based on our survey, unfortunately *none* of the existing FL methods are able to achieve these goals altogether. For instance, sharing neighborhood information through Fully Homomorphic Encryption (FHE) could mitigate the accuracy loss on the non-IID clients' graphs [14], but doing so will introduce prominent overhead. Clipping the clients' contributions can curb the poisoning attack [15], but the training convergence and model accuracy will be affected adversely [16].

Is it possible to build a federated GNIDS that achieves effectiveness, scalability and robustness all together, rather than sacrificing one goal for another?

We answer firmly to this question by designing a new Federated Graph Learning (FGL) protocol for GNIDS. 1) We found that by sharing a small piece of clients' information, i.e., the aggregated node number, the central parameter server can effectively initialize the clients' initial weights, and mitigate the impact of non-IID clients. Such information is often already accessible within an organization so no extra privacy leakage will be introduced. We instantiate this idea with a new FL bootstrapping stage based on *reference graph synthesis* and *graph sketching*, which only involve lightweight computation on the parameter server and FL clients. 2) We found that each client can self-adjust its contribution based on the divergence between client-to-global models, and we developed a new technique termed *adaptive contribution scaling (ACS)* to instantiate this idea. 3) The attacker needs to scale up the model updates to effectively poison the trained global model. Since the clients' weights are adjusted under ACS already, we can bound the model updates by tweaking ACS. Interestingly, such a combination enables *dynamic clipping*, which can address the limitation of static clipping [16]. Through theoretical analysis, we *formally prove* that the iteration-wise difference shifting is bounded under ACS, and convergence rate is still bounded. This new theoretical result suggests our protocol could be useful for other FGL applications aiming to achieve robustness on non-IID clients.

Evaluation of ENTENTE. We conduct an extensive evaluation on ENTENTE, focusing on its effectiveness in detecting abnormal interactions between entities. We adapt ENTENTE to two exemplar GNIDS, namely Euler [4] and Jbeil [5], as they embody quite different designs. We choose three real-world large-scale log datasets, OpTC [17], LANL Cyber1 (or LANL) [18] and Pivoting [19] for evaluation and simulate different client numbers. In summary, ENTENTE can boost the performance of both GNIDS models on all the datasets. On OpTC, ENTENTE outperforms all the other baseline FL methods and *even the non-FL version* (the GNIDS is trained using all data) with over 0.1 increase of average precision (AP). On LANL and Pivoting, when link prediction is conducted by Jbeil, high AP and AUC can be reached by ENTENTE (over 0.9 in most cases), for both transductive learning and inductive learning modes. On LANL when Euler is used, given only hundreds of redteam events are used for edge classification, the AP is low for all FL methods, but ENTENTE still outperforms the other methods in most cases.

We also consider the robustness of ENTENTE under adaptive attacks and consider model poisoning [20] as the main threat. We develop a *new* poisoning attack that scales up the model updates [20] and adds covering edges [21] to the GNIDS setting, by replaying malicious edges from the testing period to the training period. Since ENTENTE integrates norm bounding when adjusting clients' weights, the attack success rate is bounded to a very low rate, e.g., less than 10% when attacking Euler+LANL. Without norm bounding, not only does attack success rate increase, but the FL training process might not even finish when the attacker scales the model updates by a very large ratio. Overall, our study shows promise in addressing the data sharing concerns in building GNIDS in practice.

Contributions. We summarize the contributions of this paper as follows[1]:

- We propose a new system ENTENTE that can train a GNIDS model without requesting data sharing among departments/ organizations, under the framework of FL.
- We address the threats like non-IID client graphs and adaptive attackers with novel techniques like reference graph synthesis, graph sketching and adaptive contribution scaling. We also formally prove that the convergence rate is bounded.
- We conduct the extensive evaluation using large-scale log datasets (LANL, OpTC and Pivoting), and the result shows ENTENTE can outperform other baselines in most cases.
- We release the code at https://github.com/uci-dsp-lab/ENTENTE.

II. BACKGROUND

A. Graph-based Network Intrusion Detection Systems (GNIDS)

Network logs collected by network appliances like firewalls and proxies have been extensively leveraged to detect various cyber-attacks, including APT attacks [1]. Many graph-based approaches have been developed in recent years and we term them Graph-based Network Intrusion Detection Systems

[1]The extended version of this paper is available at arxiv [22].

(GNIDS). At the high level, for each log entry, the GNIDS extracts the subject and object fields (e.g., host) as nodes, and fills the edge attributes using the other fields (e.g., the instruction contained in the network packet). In Figure 1, we illustrate an example of a graph generated from network logs collected by different organizations (or clients).

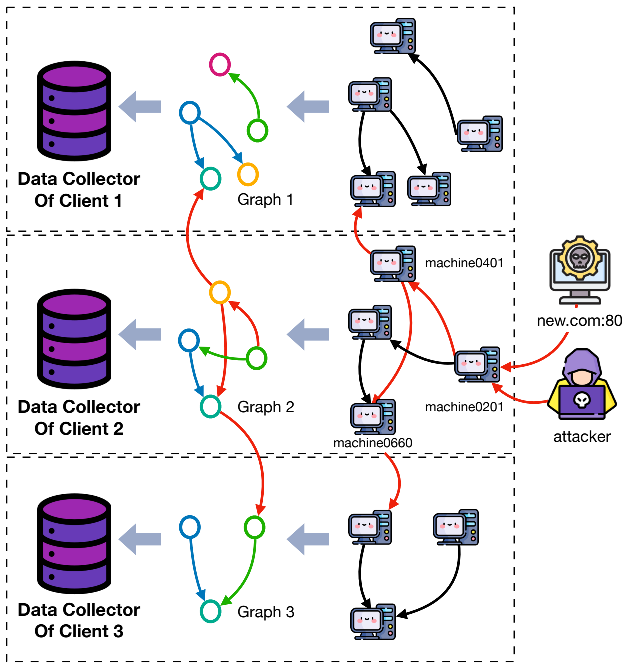

Data Collector Of Client 1
Graph 1
machine0401
new.com:80
Data Collector Of Client 2
Graph 2
machine0201
machine0660
attacker
Data Collector Of Client 3
Graph 3

Fig. 1. An example summarized from the day 1 attack campaign in the OpTC dataset [17]. The attacker first connects to `machine0201` and downloads the PowerShell attack tool. Then it pivots to machine `machine0401` and `machine0660` with the Windows WMI command. Finally, the attack spreads to other machines. The attacked machines can belong to multiple organizations (or clients). Graphs can be constructed separately from the logs collected by different clients.

On top of the graph data, GNIDS can perform anomaly detection with a trained model. The relevant works can be divided by their classification targets: sub-graphs (e.g., a graph snapshot), nodes (e.g., a host) and edges (e.g., interactions between hosts). In order to accurately model the patterns in the graph data, most works choose a Graph Neural Network (GNN). One prominent technique is *graph autoencoder (GAE)* [7], which uses a *graph encoder* to generate node embedding and a *graph decoder* to reconstruct a similar graph. The downstream tasks like edge classification can be done by generating edge scores from node embedding and comparing them with a threshold. In Section IV, we elaborate the common design choices of GNIDS models when describing the workflow of ENTENTE.

Noticeably, a relevant line of research is provenance- or host-based intrusion detection system (PIDS or HIDS) [23], [24], which detects intrusions on the *host log graph*. In Section VII, we discuss this line and the potential changes to our system ENTENTE for adaptation.

B. Federated Learning

Federated Learning (FL) is an emerging technique to allow multiple clients to train a model without revealing their private data [25], [26]. FL relies on a central parameter server to train a global model under multiple iterations. At the start of each iteration i, the server transmits a global model (w_i) to a set of clients $(1, \ldots, K)$ and they train local models $(w_i^1, \ldots, w_i^K)$ from w_i. Then the clients transmit the local models to the server to average their differences (e.g., FedAVG on model parameters [26]) and generate a new w_{i+1}.

FL has two main deployment settings: *horizontal FL* and *vertical FL* [13]. For horizontal FL, the clients' datasets have a large overlap in the same feature space but little overlap in the sample space (e.g., every client owns the same type of network logs of its sub-network machines). On the contrary, vertical FL assumes the clients have a large overlap in the sample space but little overlap in the feature space (e.g., each client collects a unique type of logs for all organizational machines). In this work, we focus on horizontal FL, which has been studied more often [13]. We also focus on the *cross-silo* FL setting, in which a small number of *organizations* participate in the *entire* training process [27], rather than the cross-device FL setting, in which many user-owned devices selectively participate in different training iterations.

Federated Graph Learning (FGL). Initially, FL was developed for tasks related to Euclidean data like image classification [26]. Recently, FL has been applied to non-Euclidean data like graphs [28], [29], and these works are termed under *Federated Graph Learning (FGL)*. Under the horizontal FL setting, the graph data is partitioned across clients, where each client has a sub-graph with non-overlapping (or little overlapping) nodes. A prominent challenge for sub-graph FL is the heterogeneity between clients' subgraphs, such that the sizes and topology are vastly different between subgraphs. While there are general solutions to address the data heterogeneity issues under FL [30], [31], some solutions are customized to sub-graph FL [32], [33]. For example, FedGTA proposed a topology-aware optimization strategy for FGL [32], but it requires heavy changes on the design of existing graphical models.

Alternatively, some works propose to *amend* each subgraph with some information shared by other clients or server [34], [35], [36]. For example, the server in FedGL asks the clients to upload node embeddings to generate a global pseudo embedding [34]. FedSage+ asks the clients to train a neighborhood generator jointly [35]. However, there is no guarantee that the shared information will not leak more clients' information (e.g., node embedding can lead to inference attacks [37]). To mitigate the privacy concerns, cryptographic primitives and/or differential privacy have been tested to amend the subgraph in a privacy-preserving way [14], [35]. For example, FedGCN allows a client to collect 1-hop or 2-hop averaged neighbor node features from clients with Fully Homomorphic Encryption (FHE) [14]. However, significant computation and communication overhead will be incurred. In this work, we develop a new FGL technique to tackle the subgraph heterogeneity issue and apply FL to GNIDS in practice.

III. Overview

In this section, we first describe the deployment settings of our system ENTENTE. Then, we demonstrate the goals to be achieved by ENTENTE. Finally, we describe the threat model.

A. Deployment Settings

We assume an organization consists of multiple sub-organizations, but it is not always feasible for them to share the raw logs with each other, e.g., when they are located in disjoint regions that are governed by privacy laws like GDPR, as elaborated in Section I. Each sub-organization collects the logs about the network packets sent to and received by its controlled machines, with systems like SIEM [38], and trains a local GNIDS model to detect past or ongoing attacks by analyzing the logs. To achieve better detection coverage and effectiveness, they decide to perform FL to jointly train a global GNIDS model that can be used by each participating sub-organization. The same procedure can be taken by multiple independent organizations to train a global GNIDS model.

Here we formally define the entities that deploy our system. Figure 1 also illustrates the setup.

- A **client** is the sub-organization that collects logs from its managed machines and trains a GNIDS model to perform intrusion detection.
- A **parameter server** is operated by an entity outside the clients (e.g., the parent organization of the clients) to aggregate the clients' model updates and push the global model to clients.
- A **machine** owned by a client is subjected to attacks. It produces network logs that are collected by the client. Each machine is also called a **node** under the client graph.

B. Design Goals and Challenges

When designing ENTENTE, we identify several goals that should be achieved to enable its real-world deployment.

- **Effectiveness.** ENTENTE should achieve high detection accuracy and precision on large-scale real-world logs. Achieving high precision is more important due to the imbalanced data distribution in the log dataset [39]. ENTENTE should achieve comparable effectiveness as the GNIDS trained on the entire log dataset.
- **Scalability.** The introduced FL mechanisms should be scalable when training a global model from large-scale log datasets owned by many clients. The communication overhead and latency added by ENTENTE on each client should be small.
- **Robustness.** In addition to compromising the client machines, the attacker has the motivation to compromise the FL procedure. ENTENTE should be able to defend against such adaptive attacks.

Challenges and solutions. The major challenge is the heterogeneity among clients. Previous studies have discovered that when clients' data are non-IID (independent and identically distributed), the effectiveness and robustness of the trained global model can be significantly degraded [40]. In our case, it is very likely that each client has divergent subgraphs in terms of size and topology. As supporting evidence, Dambra et al. studied the malware encounters using the telemetry data from a security company, and it shows enterprises in the United States have more than 5x monitored end-hosts than any other country [41].

We found that when clients' data are non-IID graphs, none of the prior FL (e.g., FedAvg) or FGL (e.g., FedGCN) approaches can achieve the aforementioned goals altogether, as they have to *sacrifice one goal for another*. For example, FedAvg is highly scalable but performs poorly under non-IID data [40]. FedGCN [14] aims to address the effectiveness challenge on the heterogeneous client data with heavyweight cryptographic methods, which sacrifices the system scalability. Norm bounding [15] improves the robustness by clipping abnormal model updates, but prior research also demonstrated it will slow down the convergence of FL training and decrease the effectiveness of the trained model when a static clipping threshold is used [16].

Hence, new FGL methods are desired in our setting, and our key observation is that by sharing a small piece of information from clients to the parameter server, i.e., the aggregated amount of nodes, the parameter server can adjust the clients' contributions automatically to offset the impact of non-IID graphs. Besides, a client can perform self-adjustment of its contribution based on the divergence of client-to-global model parameters. Since the contributions are dynamically adjusted, the limitation of defenses like norm bounding can be remedied. Overall, through new FGL protocols that are carefully designed around non-IID client graphs, our system ENTENTE achieves the three design goals *all together* for the first time.

C. Threat Model

First, we follow the threat model of the other GNIDS (e.g., [4], [5]) that though the machines can be compromised, the network communications are correctly logged by the network appliances. Hence, log integrity can be achieved. Though it is possible that advanced attackers could violate this assumption, additional defenses (e.g., using Trusted Execution Environment) can be deployed as a countermeasure [42], [43], [44].

Second, we assume the central parameter server is honest-but-curious, which is trustworthy for aggregation but may be curious about clients' local data (e.g., communications between two employees of a sub-organization). To mitigate privacy leakage, we only allow the parameter server to know the total number of nodes aggregated from all clients, in addition to the clients' model updates required by FL. We argue that the total node number is often accessible within an organization: e.g., an administrator can get the list of users across sub-organizations from the central active directory server [45]), so no extra privacy leakage is expected. In Section VI, we discuss the privacy threats potentially caused by deploying ENTENTE, including membership inference attack and gradient inversion attack, and provide preliminary analysis under the lens of differential privacy (DP).

Finally, we consider the attacker who is able to launch the poisoning attack by controlling one or more FL clients. The controlled clients are subject to data poisoning [20] (e.g., the adversary commands some compromised machines to initiate covering communications during the training stage) or model poisoning [20] (e.g., the adversary manipulates the updates of local models). In Section V-D, we introduce a concrete attack against GNIDS in the FL setting, and demonstrate ENTENTE is an effective defense.

IV. DESIGN OF ENTENTE

In this section, we describe ENTENTE in detail. ENTENTE encompasses GNIDS components that are adapted from the existing works and FL components that train the GNIDS models. We highlight FL-related components with "♦" as they are the key contributions of this work. The high-level workflow of ENTENTE is illustrated in Figure 2. The main symbols and their meanings are summarized in Table I.

TABLE I
THE MAIN SYMBOLS USED IN THE PAPER.

Term(s)	Symbol(s)
Client number, index	K, k
Client graph	$\mathcal{G}^k$
Client edges, edge	$\mathcal{E}^k$, e^k
Client nodes, node	$\mathcal{V}^k$, v^k
Snapshot number, index	T, t
Client snapshot	$\mathcal{G}_t^k$
Client model parameters	w^k
FL max and current iteration	R, i
Weight of a client update	r^k
Global model after i iterations	w_{i+1}

A. Local Graph Creation

We assume K clients have deployed the same GNIDS and they jointly train a global model with FL. Each client is indexed by $k \in [1, K]$. After the network logs are collected by client k, a graph $\mathcal{G}^k$ will be constructed by representing the event sources and destinations (machines/users/etc.) as nodes $\mathcal{V}^k$ and their communications as edges $\mathcal{E}^k$. An edge e^k between a pair of nodes v_1^k and v_2^k could contain features extracted from one or many events that have both v_1^k and v_2^k, and the commonly used features include the event frequency [4], traffic volume of network flows [3], etc. Each node has a feature vector, which can be the node type (e.g., workstation or server), privilege, etc. [4]

Though it is relatively straightforward to generate a single static graph from all events [46], such graph modeling has prominent issues like the coarse detection granularity and missing the unique temporal patterns [3]. Recent works advocate dynamic graph modeling that generates a sequence of *temporal snapshots* $[\mathcal{G}_1, \cdots, \mathcal{G}_T]$, and a snapshot $\mathcal{G}_t$ ($t \in [1, T]$) merges the events in a fix-duration time window (e.g., one hour) [3], [4], [5]. ENTENTE generates a dynamic graph by default, but it can easily be switched to static modeling by merging the nodes and edges of $[\mathcal{G}_1^k, \cdots, \mathcal{G}_T^k]$, yielding $\mathcal{G}^k$.

♦**Augmenting local graph.** Missing cross-client edges is a prominent issue for subgraph FL, and previous works tried to amend the subgraph by exchanging the topological information among clients, which however led to privacy and efficiency issues, as surveyed in Section II-B. We found that this problem can be partially addressed under the unique GNIDS deployment setting, with *1-hop graph augmentation*. Our key insight is that the log collectors deployed by a client, like firewalls, usually record the inbound and outbound cross-client events *automatically* [2]. Therefore, the cross-client edges can be harvested "for free" from each client's internal logs.

Specifically, assume $\mathcal{V}_t^k$ and $\mathcal{E}_t^k$ are nodes and edges of the snapshot $\mathcal{G}_t^k$ of client k. We search the logs to find all entities that are not contained by $\mathcal{V}_t^k$ while following the constraints of $\mathcal{V}_t^k$ (e.g., in the same snapshot period), denoted as $\mathcal{V}_t^{k\complement}$. Then $\mathcal{V}_t^{k\complement}$ will be added into $\mathcal{G}_t^k$, together with $\mathcal{E}_t^{k\complement}$ (the edges between $\mathcal{V}_t^k$ and $\mathcal{V}_t^{k\complement}$).

B. Local GNIDS Training

Node representation learning. On a generated graph snapshot $\mathcal{G}_t^k$, ENTENTE extracts the representation (or embedding) of each node with a *graph auto-encoder (GAE)* [7], which aggregates the node's features and its neighborhood information. Graph Convolutional Network (GCN) [47] is a standard choice [4], which takes the whole adjacency matrix as the input, but only transductive learning, which assumes the nodes are identical between training and testing, is supported. In our setting, the GCN encoder can be written as:

$$Z_t^k = \text{GCN}(X_t^k, A_t^k) \tag{1}$$

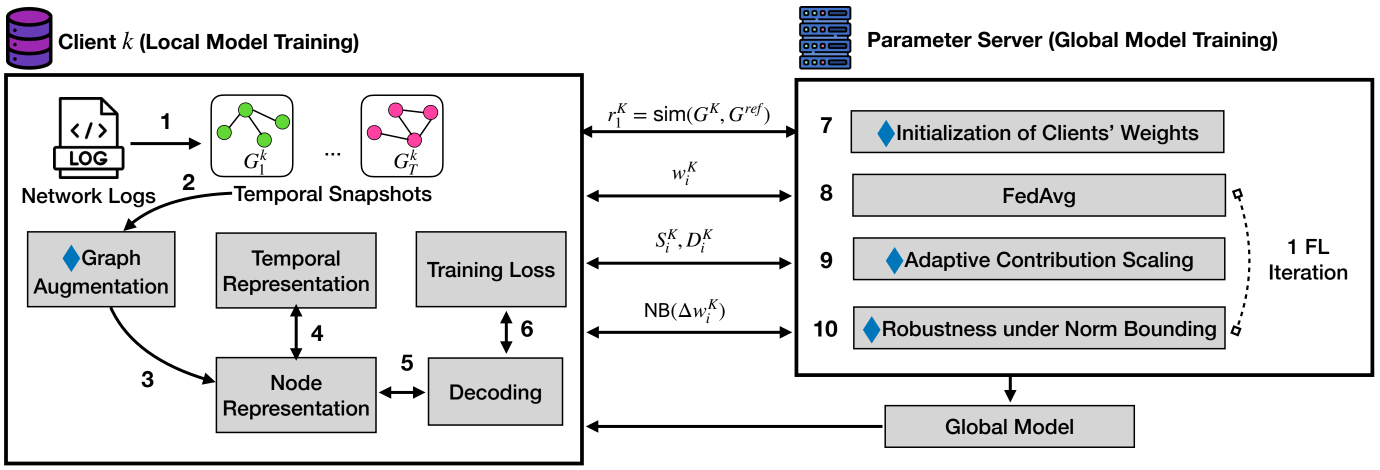
where $Z_t^k \in \mathbb{R}^{n \times r}$ is the node embedding generated by the encoder on $\mathcal{G}_t^k$, X_t^k represents the node features and A_t^k represents the adjacency matrix on $\mathcal{G}_t^k$.

To accommodate the new nodes observed in the testing phase, inductive learning has been proposed, which learns the neighborhood aggregator to generate node embeddings. GraphSage [48] is a classic encoder in this case, but recent GNIDS [5] also integrates other encoders like Temporal Graph Networks (TGN) [49]. The TGN encoder deals with a continuous-time dynamic graph, and for each time t, the node embedding Z_t^k is generated by a trainable message function, message aggregator and memory updater.

Temporal learning. The anomalous temporal patterns (e.g., a login attempt outside of work hours) provide important indicators for intrusion detection, and some recent GNIDS [4], [3] choose to capture such patterns with Recurrent Neural Network (RNN), e.g., Gated Recurrent Unit (GRU) under RNN families. The RNN module can be written as:

$$[Z_1^k, \ldots, Z_T^k] = \text{RNN}([\text{ENC}(\mathcal{G}_1^k), \ldots, \text{ENC}(\mathcal{G}_T^k)]) \tag{2}$$

where $\mathcal{G}_t^k = (X_t^k, A_t^k)$ ($t \in [1, T]$) , ENC is the encoder (e.g., GCN), and $[Z_1^k, \ldots, Z_T^k]$ are the embeddings updated by RNN after they are encoded from $[\mathcal{G}_1^k, \ldots, \mathcal{G}_T^k]$.



Client k (Local Model Training)
LOG
Network Logs
1
2
Temporal Snapshots
G_1^k
...
G_T^k
Graph Augmentation
Temporal Representation
Training Loss
4
6
3
5
Node Representation
Decoding
$r_1^K = \text{sim}(G^K, G^{ref})$
w_i^K
S_i^K, D_i^K
$\text{NB}(\Delta w_i^K)$
Parameter Server (Global Model Training)
7
Initialization of Clients' Weights
8
FedAvg
9
Adaptive Contribution Scaling
10
Robustness under Norm Bounding
1 FL Iteration
Global Model

Fig. 2. The workflow of ENTENTE. The client k trains a GNIDS model locally and communicates with the parameter server to jointly learn a global GNIDS with other clients. Function Sim computes graph similarity.

When TGN is the encoder, the temporal pattern is directly handled by its memory module, which can be a vanilla RNN and GRU [49]. Hence, no extra RNN layer is needed.

Both GCN+RNN and TGN encoders (and other encoders used by GNIDS) are supported by ENTENTE when they are trained under FL and we explain this feature in Section IV-C.

Decoding. On the node representations, the decoder aims to reconstruct the adjacency matrix with edge scores. The basic decoder is inner product decoder [4], which can be written as:

$$\text{DEC}(Z_t^k) = \Pr(A_{t+n}^k = 1|Z_t^k) = \sigma(Z_t^k Z_t^{k\intercal}) \tag{3}$$

where $\sigma(\cdot)$ is the logistic sigmoid function and $\Pr(A_{t+n} = 1|Z_t)$ is the reconstructed adjacency matrix at time $t+n$ given Z_t. The probability at each matrix cell is used as an edge score.

The simplicity of the inner product could lead to prominent reconstruction loss, and more complex, trainable decoders like Multilayer Perceptron (MLP) have been used by GNIDS [3], [5]. We design ENTENTE to support different variations of decoders as elaborated in Section IV-C.

Training loss. When training a GNIDS, the weights of trainable components, including the graph encoder, RNN and decoder, are updated under a loss function. The typical choice is cross-entropy loss, which can be written as:

$$\mathcal{L} = -\log(\Pr(A_t^k|\hat{A}_t^k)) \tag{4}$$

where $\hat{A}_t^k$ is the adjacency matrix decoded from node embeddings.

When the GNIDS is trained with only benign events in unsupervised learning mode, negative sampling [50] can be applied to randomly select non-existent data points as the malicious samples (e.g., non-existent edges [4], [3]).

C. Federated GNIDS Training

We aim to support different GNIDS graph modeling, downstream tasks, training/testing setup, as described in Section III-B. Hence, ENTENTE is designed to augment the existing GNIDS without heavy adjustment of their components, and the FGL works that redesign the local models [51], [32] are not suitable. Based on our survey of existing FL frameworks, we are motivated to build ENTENTE on top of FedAVG, because 1) it only needs clients' model parameters as input, and 2) it has demonstrated effectiveness when the clients' models are GNN [52] and RNN [53]. Since the common GNIDS components like node representation learning, temporal learning and decoding all use the same set of training data, we could train them independently with FedAvg and update their model parameters. As such, ENTENTE not only incurs minimum development overhead, but also achieves similar or even better effectiveness compared to the original GNIDS, as supported by the evaluation (Section V). We also want to point out that the standard FedAvg workflow leads to unsatisfactory performance and we elaborate on the key adjustment below.

♦**Initialization of clients' weights.** FedAVG aggregates the model parameters of GNIDS components in a number of iterations as described in Section II-B. Its basic version in one iteration can be represented as:

$$w_{i+1} = \Sigma_{k=1}^{K} r^k \times w_i^k \tag{5}$$

where i is the current iteration, w_i^k is the local model parameters of client C_k that is trained with the global model of previous iteration w_i and its local data, r^k is the weight of client k during aggregation, and w_{i+1} is the global model parameters after aggregation. As described in Section III-B, the clients' data are non-IID, which has a prominent impact on the performance of FL algorithms. To tackle non-IID data, one feasible approach is to assign different weights to the model updates from different clients, so the impact from clients with outlying distributions can be contained. Previous works have used the number of data samples (e.g., images [26]) per client for weights, but in our case, each client only has *one* subgraph.

We address this challenge with a new method to initialize the client's weights based on its *graph properties*. We minimize the communication overhead and privacy leakage by computing the client's weights on top of *a single reference graph* generated by the parameter server. This reference graph stays the same for the whole training process and across clients. Our approach is inspired by Zhao et al., which distributes a warm-up model trained with globally shared data and shows test accuracy can be increased [54].

Specifically, we assume the server knows the total number of nodes (n) from all clients, as justified in Section III-C. When bootstrapping FL, the parameter server generates a reference graph $\mathcal{G}^{ref}$ and distributes it to all clients for them to compute r^k ($\forall k \in [1, K]$). We choose to apply *Barabási–Albert (BA) Model* [55] to generate the reference graph. BA model is a *random graph model* for complex networks analysis [56], and it is selected because 1) it only needs the number of nodes (n) and the number of initial edges for a new node (m) and 2) it has low computational overhead (complexity is $O(n \times m)$) when m is small. In Section V-A and Section V-E, we discuss the selection of m and evaluate its impact. The pseudo-code of BA model is written in Appendix VIII-A.

♦**Graph sketching.** On a client C^k, r^k will be initialized based on the *graph similarity* between its generated graph $\mathcal{G}^k$ (aggregated from $[\mathcal{G}_1^k, \ldots, \mathcal{G}_T^k]$) and $\mathcal{G}^{ref}$. Intuitively, the client with a similar distribution to the global data should receive high r^k. Noticeably, we compute r^k locally on the client, so nothing about the client's data or distribution will be learned by the server. This is different from the standard FedAVG, which computes weights on the server. However, graph similarity is computationally intensive: e.g., graph edit distance (GED) is a fundamental NP-hard problem [57]. Therefore, we choose to compute the similarity on the *graph sketches* instead of raw graphs for efficiency. We use *Weisfeiler-Lehman (WL) graph kernel* [58] to capture the graph structure surrounding each node and compute *histogram* by nodes' neighborhood.

Algorithm 1: Weisfeiler-Lehman Histogram (WLH).

```
Data: Graph $\mathcal{G}$
Result: histograms
labels $\leftarrow$ InitializeLabels($\mathcal{G}$);
histograms $\leftarrow$ Histogram(labels);
for $i = 1$ to MaxIters do
    foreach $v$ in $\mathcal{G}$ do
        neigh $\leftarrow$ NeighborLabels($v$, labels);
        labels[$v$] $\leftarrow$ Hash(Sort(labels[$v$] + neigh));
    histograms $\leftarrow$
      Append(histograms, Histogram(labels))
return histograms
```

In Algorithm 1, we describe how to compute the Weisfeiler-Lehman histogram (WLH), where InitializeLabels assigns *node degree* as a label to each node v, NeighborLabels creates a multiset containing v's current label and its neighbors' labels, Hash and Sort produce a new label for v. $MaxIters$ is the number of iterations, and ith-iteration computes WLH for i-hop neighborhood. We set $MaxIters$ to 3. When applying Algorithm 1 to our setting, we compute the Jaccard similarity S^k_{Jac} between $\mathcal{G}^k$ and $\mathcal{G}^{ref}$.

$$S^k_{\text{Jac}} = \frac{\mathcal{G}^{ref} \cap \mathcal{G}^k}{\mathcal{G}^{ref} \cup \mathcal{G}^k} \tag{6}$$

♦**Adaptive contribution scaling (ACS).** We follow the FedAVG client-server protocol to update the model parameters by iterations. While FedAvg uses the same r^k throughout FL, we found the client contributions can be more precisely modeled by dynamically adjusting them towards stability. This idea at the high level has been examined by prior works like [59], [60], but we found none of them are suitable under our setting. First, they require the statistical information of local data (e.g., local gradients or local samples [59], [60]), which has been avoided by ENTENTE due to privacy concerns. Second, the contribution evaluation can be heavy (e.g., Shapley value by [61]).

Instead, we propose a *lightweight and privacy-preserving* method termed *adaptive contribution scaling (ACS)* to adjust clients' weights only using the model parameters. Specifically, for client C^k at the FL iteration i, the parameter server computes a similarity metric S^k_i based on the cosine similarity, and a distance metric based on L2 distance D^k_i. The new client' weights r^k_i will be computed with S^k_{Jac}, S^k_i and D^k_i. r^k_1 is initialized using S^k_{Jac}. ACS models both representational similarity (with S^k_i) and distance of local models to the global models (with D^k_i), hence presenting a more reliable indicator for client contributions. To mitigate model collapse incurred by some clients, the server bounds D^k_i by a hyperparameter ω. The definitions of S^k_i and D^k_i, and the new aggregation function are listed below.

$$S^k_i = \frac{||\sum w^k_i \times w_{i-1}||}{||\sum w^k_i|| \times ||w_{i-1}||} \tag{7}$$

$$D^k_i = \frac{\omega\sqrt{\sum(w^k_i - w_{i-1})^2}}{\max(\omega, \sqrt{\sum(w^k_i - w_{i-1})^2})}, \forall \omega > 0 \tag{8}$$

$$w_{i+1} = \frac{1}{K}\sum_{k=1}^{K} r^k_i \times w^k_i \quad \text{s.t. } r^k_i = c_1 \times S^k_{\text{Jac}} + c_2 \times S^k_i \times D^k_i \tag{9}$$

where c_1 and c_2 are two constants to adjust the contributions from S^k_{Jac}, S^k_i and D^k_i.

With ACS, ENTENTE is able to achieve better effectiveness in the evaluation. Moreover, we are able to *formally prove* that the iteration-wise difference shifting under Equation 9 is *bounded*, by $|\frac{w_{i+1}}{\sum_k w^k_i}| \leq c_1 + c_2\omega$ from $\sum_k w^k_i$ (FedAVG). The non-IID data can lead to slow convergence or even non-convergence in training [40]. By bounding the iteration-wise difference shifting through ACS (i.e., adjusting the e contributions from S^k_{Jac}, S^k_i and D^k_i), we are able to address this issue with a formal guarantee.

Theorem 1. *Define $S^k_{\mathsf{Jac}}, S^k_i$, and D^k_i as Equation 6, Equation 7, and Equation 8, respectively. Let c_1, c_2 be two hyper-parameters to adjust the contributions. Global model update shifting from FedAVG per iteration i is bounded by $|\frac{w_{i+1}}{\sum_k w^k_i}| \leq c_1 + c_2\omega$ for any $0 \leq S^k_{\mathsf{Jac}}, S^k_i \leq 1$.*

This analytical result shows promise that ENTENTE can be effective in other applications that use subgraph FL.

♦**Robustness via norm bounding.** Due to the use of FL, a client deploying GNIDS could be more vulnerable, when the other clients are compromised and poisoning the global model. We observe that ACS can be extended to defend against FL poisoning attack by limiting the contribution of a client, and we are motivated to integrate norm bounding (NB) [15] for this purpose. In particular, NB observes that the attacker's model updates are likely to have large norms to influence the direction of the global model. As a countermeasure, the server can bound the model updates with a threshold M to mitigate the impact of the abnormal update during aggregation. Our approach differs from the standard NB by adjusting the bounds *dynamically* under ACS.

We define $\mathsf{NB}(\Delta w^k_{i+1}) = \frac{\Delta w^k_{i+1}}{\max(1, \|\Delta w^k_{i+1}\|_2/M)}$ [15] for simplification. The global model is bounded through the updated difference Δw^k_{i+1} derived from local models as Equation 10.

$$w_{i+1} = w_i + \frac{1}{K}\sum_{k=1}^{K} r^k_i \times \mathsf{NB}(\Delta w^k_{i+1}) \quad s.t.\ \Delta w^k_{i+1} = w^k_{i+1} - w_i \tag{10}$$

Like ACS, we present Theorem 2 to confirm that ENTENTE has a bounded convergence rate for establishing a global model. We want to highlight that NB could lead to slow convergence and worse effectiveness [16] in the sacrifice of robustness, but such an issue is well resolved when combining ACS and NB, achieving *dynamic clipping*. In Appendix VIII-E, we further extend Equation 10 to support differential privacy (DP), when the privacy of model output is a concern. Overall, our method entails a strong guarantee that effectiveness and robustness can be preserved concurrently on non-IID graph data.

Theorem 2. *Assume all Lemmas 1,2,3,4 and constrains [62] reviewed in Appendix VIII-F. For any bounding norm $M \geq \eta EG$, theoretical complexity of* ENTENTE*'s convergence is $O(1/(R\eta E(c_1 + c_2\omega))) + O((c_1+c_2\omega)\eta/K \cdot \sigma^2_{\mathsf{local}}) + O(\eta^2 E^2 \cdot \sigma^2_{\mathsf{global}})$.*

D. Intrusion Detection

After the model is trained, for intrusion detection by GNIDS, different classification granularities can be applied. Edge classification compares the edge scores generated from the GNIDS decoder with a threshold, which can be learnt from validation snapshots, and achieves the finest detection granularity [4], [3], [5]. Node classification adds a classification layer (e.g., softmax) on top of the node embeddings and compares the probabilities with a threshold [5]. Alternatively, one can compute a score for the whole snapshot by aggregating the edge scores and detect the abnormal ones. In this work, we test ENTENTE over edge-level classification due to its finest detection granularity. Supporting other detection granularities is trivial by applying the aforementioned changes.

In Appendix VIII-B, we also summarize the detailed workflow of ENTENTE in pseudo-code.

TABLE II
THE STATISTICS OF THE THREE TESTED DATASETS.

Dataset	#Nodes	#Events	Duration
OpTC	814	92,073,717	8 days
OpTC-redteam	28	21,784	3 days
LANL	17,649	1,051,430,459	58 days
LANL-redteam	305	749	28 days
Pivoting	1,015	74,551,643	1 day

V. EVALUATION

This section describes our experiment setting, including the evaluated GNIDS, datasets, baseline FL methods, etc. Then, we consider the effectiveness of ENTENTE and other baseline FL methods on different combinations of GNIDS and datasets. Next, we show that ENTENTE fulfills the scalability and robustness goals. An ablation study is performed in the end to understand the impact of individual components and hyper-parameters.

A. Experiment Settings

Evaluated GNIDS. We adapt two GNIDS models Euler [4] and Jbeil [5] under ENTENTE. We chose them because they have open-source implementations and both have been tested on large-scale network datasets. In addition, their architectures are very different, so we can assess whether the design goals can be achieved across different GNIDS modes. For instance, due to the use of GCN, Euler only supports transductive learning, while TGN used by Jbeil supports both transductive learning and inductive learning. Both learning modes are tested by us.

Datasets. We use OpTC, LANL cyber1 (or LANL for short) and Pivoting datasets to evaluate ENTENTE. These datasets have been used by our baseline GNIDS [4], [5] and thoroughly tested by other GNIDS like [3], [63]. In Appendix VIII-C, we describe the characteristics of the datasets and how we preprocess them. Table II shows their statistics.

All datasets are highly non-IID as reflected in Section V-E. For LANL and OPTC, we observe high standard deviation on node number in different clients, and in Pivoting, the event numbers of clients have high standard deviation. Such a dataset characteristic justifies the design of ENTENTE.

Data split for clients. To simulate the FL process, we need each client to keep a subgraph of the complete graph. The prior FGL studies choose to cluster the nodes and generate local graphs [14], [35]. We follow this direction and leverage a recent approach Multilayer Block Model (MBM) [64] to cluster the logs. MBM clusters system events to visualize the major clusters and major events between different clusters. For LANL, MBM has built-in support and we use its code to generate a user-machine bipartite graph and change its "Number of bottom clusters" to get different numbers of sub-graphs [65]. For OpTC, we categorize its hosts into internal and external nodes, like how MBM processes the VAST dataset (a dataset with simulated network traffic), and generate sub-graphs like processing LANL. For Pivoting, we follow similar settings as LANL except that we replace machines with hosts, and use the protocol and destination port to fill the node type.

Metrics. For Euler, we define true positive (TP) as a malicious edge in a snapshot that contains at least one redteam event, and true negative (TN) as a normal edge without any redteam event. False positive (FP) and false negative (FN) are misclassified as malicious and normal edges. For Jbeil, we consider TP as a non-existent edge and TN as an existing edge while FPs and FNs are misclassified as non-existent and existing edges With TP, TN, FP and FN defined, we compute precision, recall and FPR as $\frac{TP}{TP+FP}$, $\frac{TP}{TP+FN}$ and $\frac{FP}{TP+FP}$.

The values of precision, recall and FPR depend on the classification threshold, which might not always be optimal, e.g., when the validation dataset has a different distribution from the testing dataset. Hence, we also compute the area under the ROC curve (AUC) which is computed over all thresholds. When the malicious and normal edges are highly imbalanced, AUC might not correctly capture the effectiveness of a GNIDS, as it measures the relation between TPR and FPR (see "Base Rate Fallacy" in [39]). So, we also compute average precision (AP), which is defined as:

$$\text{AP} = \sum_n (R_n - R_{n-1}) \times P_n \tag{11}$$

where R_n and P_n are the precision and recall at the n-th threshold. It measures the area under the precision-recall curve, which "conveys the true performance" [39] on an imbalanced dataset. We use AUC and AP as the major metrics in the percentage format following [66], [67].

Baselines. We consider 5 baseline models to compare with ENTENTE. Except for Non-FL, all the other models are well-known FL models or extensions. For them, the cross-client edges have been added to the local graphs for a fair comparison with ENTENTE.

- **Non-FL.** For this method, we assume GNIDS is directly trained on the whole training dataset and FL is not involved.
- **FedAVG [26].** We use the basic version and use the same weight for each client.
- **FedOpt [30].** FedOpt seeks to improve the convergence and stability of FL in non-IID settings. It employs adaptive local solvers and a server-side momentum term to achieve faster convergence.
- **FedProx [31].** Similar to FedOpt, FedProx is designed to handle non-IID settings. It introduces a proximal term to the local optimization objective of each client, which helps prevent divergence when local datasets are skewed.
- **FedAVG-N.** This is a simple extension of FedAVG, in which a client's weight depends on its node numbers n^k, i.e., $r^k = \frac{n^k}{n}$, where n is the total node number. r^k is constant across iterations.

To evaluate the impact of norm bounding on GNIDS utility, we create a variation of ENTENTE, termed ENTENTE-UB, that performs without Equation 9.

Hyper-parameters. We set $m = 5$ for Barabási–Albert (BA) model. For the client numbers K, we vary it from 2-5 for LANL, 2-5 OpTC, and 2-4 for Pivoting. We found if we further increase K, MBM will yield invalid clusters (e.g., empty clusters). For Pivoting, because the edge features useful for clustering are fewer (only the port number can be used), generating clusters with sufficient node numbers from $K \geq 5$ is infeasible. Yet, for the scalability analysis in Section V-C, we tried $K = 10, 20$ for LANL by dropping invalid clusters. The number of local epochs E is set to 1, except for FedOpt, for which we set E to 5. For FedProx, we set its parameter μ, which controls the weight of the proximal term, to 0.05. For the learning rate η, we set 0.01 for LANL, 0.005 for OpTC and 0.003 for Pivoting. We use Adam Optimizer. For ACS hyper-parameters, we set a large c_1 as 0.8 and a small c_2 as 0.2 to avoid drastic changes on w_{i+1}. ω is set as 5. For norm bounding parameter M, we use 5 for Euler and 70 for Jbeil. We use a larger value of M for Jbeil because the norm w of Jbeil is empirically large.

Environment. We run the experiments on a workstation that has an AMD Ryzen Threadripper 3970X 32-core Processor and 256 GB CPU memory. The OS is Ubuntu 20.04.2 LTS. We use PyTorch 1.10 and Python 3.9.12 as the environment when building ENTENTE. We use GPU implementations as default and our GPU is NVIDIA GeForce RTX 3090 with 24GB memory. For Euler, we use its source code from [68] and disable its distributed setting so FL can be implemented. For Jbeil, we use its source code from [69].

B. Effectiveness

We evaluate the overall performance of ENTENTE under the default hyper-parameters and compare the results with the other baselines. We list the results when $K = 3 - 5$ and the results under $K = 2$ are shown in the Appendix VIII-D. We assume no poisoning attacks happen here and leave the evaluation under poisoning attacks in Section V-D.

Results on OpTC. In Table III, we compare the AUC and AP of different FL methods, with different client numbers K. First, we found ENTENTE and ENTENTE-UB clearly outperform the other FL methods for *every* K, reaching 74%-84% AP and over 99% AUC. The accuracy loss due to norm bounding is also acceptable. Simply initializing the clients' weights with node numbers (FedAVG-N) boosts the performance of FedAvg, but not yet reached the same level of ENTENTE (e.g., 69.96% AP vs 83.29% AP under $K = 3$). Though FedOpt and FedProx are designed to address the non-IID issue of FedAVG, their performance is even worse than the simple variation of FedAvg (FedAVG-N), suggesting pure data-adaptive tuning is difficult on imbalanced datasets.

Interestingly, we found that ENTENTE even outperformed Non-FL, which trains the GNIDS using all data. In fact,

TABLE III

EVALUATION ON EULER AND JBEIL UNDER DIFFERENT CLIENT NUMBERS K. NON-FL IS LISTED FOR EASE OF COMPARISON. ALL NUMBERS ARE USED IN THE PERCENT FORMAT. THE BEST AND THE SECOND-BEST AMONG FL METHODS ARE HIGHLIGHTED RESPECTIVELY. ⬆MEANS OUTPERFORMING NON-FL AND ▼MEANS THE WORST PERFORMANCE.

	OPTC-Euler								LANL-Euler							
Client#	2		3		4		5		2		3		4		5	
Algorithm	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC
Non-FL	69.96	98.76	69.96	98.76	69.96	98.76	69.96	98.76	0.89	97.93	0.89	97.93	0.89	97.93	0.89	97.93
FedAVG	33.69▼	95.78▼	65.72▼	98.62▼	66.41▼	97.00▼	75.19⬆	98.66	0.71	98.08⬆	0.10	98.08⬆	0.22	98.41⬆	0.57	98.79⬆
FedOpt	66.62	98.37	67.24	99.19⬆	70.09⬆	98.26	72.19⬆	98.84⬆	0.65	97.87	0.60	98.80⬆	0.60	99.08⬆	0.01▼	84.45
FedProx	64.29	98.07	72.46⬆	99.25⬆	73.79⬆	97.30	71.38⬆	98.25▼	0.72	97.90	0.01▼	81.98▼	0.19▼	98.84⬆	0.01▼	81.64▼
FedAVG-N	70.41⬆	99.13⬆	69.91	99.29⬆	78.40⬆	98.97⬆	71.87⬆	98.78⬆	0.51▼	96.85▼	0.62	97.97⬆	0.41	96.80▼	0.53	96.68
ENTENTE-UB	77.14⬆	99.50⬆	82.95⬆	99.68⬆	84.96⬆	99.69⬆	78.12⬆	99.24⬆	0.70	97.07	0.65	97.95⬆	0.82	97.01	0.65	97.33
ENTENTE	74.60⬆	99.41⬆	83.29⬆	99.76⬆	80.82⬆	99.79⬆	82.03⬆	99.79⬆	0.77	98.92⬆	0.87	97.68	0.72	97.00	0.67	97.16

	LANL-Jbeil-Transductive								LANL-Jbeil-Inductive							
Client#	2		3		4		5		2		3		4		5	
Algorithm	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC
Non-FL	96.47	97.51	96.47	97.51	96.47	97.51	96.47	97.51	94.47	96.21	94.47	96.21	94.47	96.21	94.47	96.21
FedAVG	88.54	91.85	66.77▼	72.98▼	50.94▼	51.84▼	59.64▼	65.11▼	87.30	91.34	67.25▼	73.89▼	50.06▼	50.12▼	58.77▼	63.86▼
FedOpt	59.67▼	64.64▼	82.18	85.81	60.75	67.35	72.58	80.06	58.79▼	63.50▼	82.00	86.12	62.19	69.20	75.16	82.33
FedProx	61.00	67.71	67.29	74.32	70.89	77.87	76.52	82.02	62.80	69.94	68.28	75.29	70.84	77.96	76.52	81.88
FedAVG-N	71.37	77.46	88.96	91.14	61.92	68.93	63.30	70.16	67.33	74.13	90.50	92.58	63.85	71.31	62.80	69.77
ENTENTE-UB	95.40	96.47	77.47	82.98	90.29	91.47	85.46	86.22	95.54⬆	96.55⬆	82.12	86.99	91.28	92.65	88.09	88.83
ENTENTE	94.43	96.25	87.47	88.80	84.10	85.09	81.05	87.70	93.58	95.88	89.13	90.49	87.80	86.36	80.56	87.48

the AP achieved by ENTENTE is 5%-14% higher compared to Non-FL. This might seem counter-intuitive, but a similar observation was also documented in a recent work [70], which shows that if class imbalance and data heterogeneity are well handled, FL methods can achieve better results than non-FL training on the global long-tailed data.

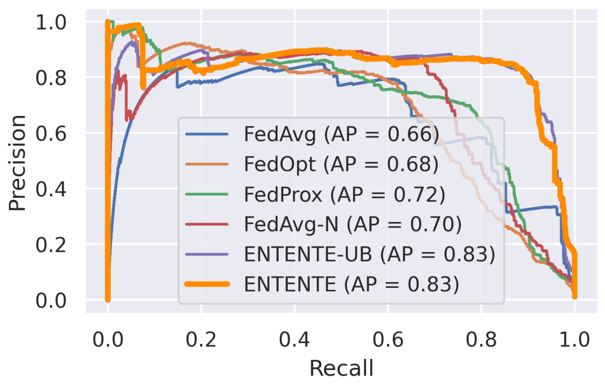


Fig. 3. The precision-recall curve under different FL methods on OpTC dataset ($K = 3$).

In Figure 3, we draw the precision-recall curve of different FL methods with one client number ($K = 3$) due to space limit. The precision of ENTENTE outperforms the other systems at most recall values, and it can reach 0.8 precision at about 0.9 recall.

Results on LANL. In Table III, we show AP and AUC when LANL is tested with Euler. Since Euler uses the redteam events as TP, the class distribution is highly imbalanced, and all systems have low AP[2]. ENTENTE and ENTENTE-UB do not outperform non-FL but are still the ones closest to non-FL in all cases ($K = 2 - 5$) in terms of AP. We notice that the trend of AP is somehow opposite to AUC (e.g., AP of ENTENTE is higher than FedAvg when $K = 3, 5$) and this observation can be attributed to the differences in the training and testing data distribution. In particular, Euler uses negative sampling to "synthesize" malicious edges during training. However, the ground-truth malicious edges during testing have a much smaller volume (only hundreds), which might not be characterized well by the sampled negative edges. Correctly classifying malicious edges is a major goal modeled under AP, but not necessarily so under AUC, when the vast majority of edges are benign. We acknowledge that the AP is still low, and discuss this limitation in Section VI. Yet, we argue that the *improvement over the existing systems* is more important, and ENTENTE achieves this goal.

[2]The paper of Euler [4] claims 5.23% (GCN+GRU models) AP on LANL, but we cannot reproduce the same result with their GitHub code [68]. A few possible explanations include we use different random seeds and different hardware (we use GPUs while Euler uses CPUs).

Table III also shows the result when Jbeil is the GNIDS. This time, as non-existent edges are considered as TP (explained in Section V-A), higher AP is observed. We found ENTENTE still outperforms the other baselines (except $K = 3$, lower than FedAVG-N). Though ENTENTE performs worse than ENTENTE-UB, the margin is small. The more balanced class distribution makes non-FL the winner in most cases.

Results on Pivoting. Due to space limit, we present the results in Appendix VIII-D. In short, we found ENTENTE achieves the best AP and AUC in the majority of the cases among the FL methods.

C. Scalability

Following the scalability requirement described in Section III-B, we measure the communication overhead, latency and memory consumption caused by ENTENTE.

We first follow the default data split setup ($K = 2 - 5$). Regarding latency, when training LANL and OpTC with Euler, the whole training process takes an average of 116.33 seconds and 706.40 seconds (30 FL iterations till the training converges for LANL and OpTC) and the testing process takes

160.99 seconds and 0.6139 seconds[3]. For Pivoting, the training process takes an average of 314.58 seconds (5 FL iterations) and the testing process takes 45.963 seconds.

Compared to basic FedAVG, ENTENTE introduces additional overhead from generating the global reference graph and updating client weights using WLH and ACS. On LANL with $K = 4$, the first step takes 5.47, 7.90, 2.70, and 0.24 seconds across the four clients, while the second step takes only 0.082 seconds. On OpTC, the first step takes 0.54, 2.25, 0.89, and 25.68 seconds, and the second step takes 0.0058 seconds. On Pivoting, the first step takes 0.049, 0.009, 0.082, and 0.035 seconds, and the second step takes 0.051 seconds. The differences between clients are due to their different sizes. Overall, while ENTENTE adds some computation time, the overhead remains minor relative to the full training process.

By using FL, ENTENTE also reduces the data transmitted from client to server when training a GNIDS model, saving clients' bandwidth. Take Euler+LANL as an example. The central server would need to access the 65GB network logs to train GNIDS. With ENTENTE, only model weights w_i^k are transmitted by a client k in an FL iteration i, and only *1.94MB* model weights are transmitted in total. Then, we estimate the per-iteration latency for LANL+Euler ($K = 4$) considering network transmission overhead. Assuming the client upload link bandwidth is 1 MB/s following the assumption of the FedAvg paper [26] and the download link bandwidth is 10 MB/s. The server has much larger bandwidth and it will not throttle clients' network transmission. The max client computation time per iteration is 7 s. The latency per iteration will be $\frac{1.94}{1} + \frac{1.94}{10} + 7 = 9.134s$. The latency is expected to be similar for larger K if server bandwidth is much larger than client's.

TABLE IV
PERFORMANCE OF LANL+EULER UNDER LARGE K (FROM 5 TO 50). "TRAINING" IS THE LATENCY OF THE WHOLE TRAINING PROCESS. "CPU" AND "GPU" ARE THE AVERAGED CPU AND GPU MEMORY OVERHEAD PER CLIENT.

K	Training(s)	AP(%)	AUC(%)	CPU(MB)	GPU(MB)
5	138.51	0.67	97.16	1521.21	3343.39
10	224.38	0.77	98.72	866.13	3256.13
20 [4]	497.46	1.05	99.12	522.87	3308.80
50 [5]	5111.43	0.20	97.99	1341.30	N/A

Finally, we attempt to increase K to be more than 5 and test whether ENTENTE can scale up to many clients, and show the results in Table IV. We evaluate this setting on the LANL dataset because it contains a large number of nodes (17,649). For OpTC and Pivoting datasets, increasing K will yield invalid clusters from MBM as discussed in Section V-A. Yet, for large K (e.g., 10, 20, and 50), we found MBM generates empty clusters, which impedes generating more valid clients, so we treat the non-empty clusters as clients (9, 15, 35 for $K = \{10, 20, 50\}$). We found that the training time does increase notably when $K = 9$ and $K = 15$, but this is because the data loading time is longer (more cross-client edges appear when K increases). Besides, since we simulate all clients on a single machine and they share a GPU, the resource contention also elongates the training time. The training time is expected to reduce when the clients operate in a real-world distributed environment. With $K = 50$, due to GPU memory limitation, we only test on the CPU environment. AP and AUC are also recorded, and we find that AP and AUC get better when K is set to medium values and drop again when K is increased to 50. We further measure memory consumption introduced by ENTENTE for a larger number of clients to assess its per-client overhead. For each run of ENTENTE, we measure CPU and GPU overhead separately and compute the average consumption across clients. Specifically, the CPU memory overhead per client drops from 1521 MB to 522 MB when K rises from 5 to 20, which is caused by the reduction of the graph size processed by each client. For $K = 50$, the average CPU usage is smaller compared to $K = 5$ (1521.21 MB vs 1341.30 MB). We speculate the overhead is mainly caused by the CPU–GPU transfer buffers and data loading since only the CPU is used in the setting. GPU memory usage remains relatively stable (around 3.2–3.3 GB), suggesting that the GPU memory footprint is largely determined by the model architecture.

D. Robustness against Poisoning Attacks

In this subsection, we evaluate how ENTENTE performs when FL poisoning attack is conducted by a compromised client. Recently, a few adaptive attacks against GNIDS were developed [21], [71], [72]. Only Xu et al. [21] tested the training-time attacks (the other focused on testing-time attacks). So far, none of the prior works considered the robustness of a GNIDS model trained under FL, and we adapt the attack from Xu et al. [21] to our setting, which has been evaluated against Euler on the LANL dataset.

In the original attack of Xu et al. [21], *covering* accesses are added for each malicious edge to avoid exposing the malicious edges when GNIDS is trained. In our setting, the attacker can directly inject the malicious edges to the logs *after* they are collected by the FL client, under the model poisoning adversary [20]. As such, the covering edges are unnecessary. In addition, the attacker can scale up the model updates by a constant factor γ to outweigh the updates from other benign clients [20].

We present the attack pseudo-code in Algorithm 2. Specifically, on a controlled client k, the attacker needs to enumerate all the sub-graphs $[\mathcal{G}_1^k, \dots, \mathcal{G}_T^k]$ and compare the source and destination nodes to its malicious edges $\mathcal{EM}$ to be used in the testing time. An edge will be added when there is a pair-wise match. We also use a likelihood threshold p to control the number of injected edges.

[3]OpTC has much larger training latency and much smaller testing latency than LANL because only 23% of whole OpTC data is used for testing (while 96% for LANL).

[4]With $K = 20$, the clustering process results in only 15 valid, non-empty clients.

[5]With $K = 50$, the clustering process results in only 35 valid, non-empty clients. This experiment was executed on a CPU due to GPU memory limitations.

Algorithm 2: The proposed poisoning attack. ClientUpdate is the same as Algorithm 4. Each client's graph $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{E}^k)$

```
Data: Number of subgraphs $T$; Malicious edges $\mathcal{EM}$;
      Likelihood threshold $p$
foreach client $k$ from the malicious clients do
    for $t \leftarrow 1$ to $T$ do
        if random number $< p$ then
            for $e \in \mathcal{EM}^k$ do
                if $e \notin \mathcal{E}_t^k$ and $e.src \in \mathcal{V}_t^k$ and $e.dst \in \mathcal{V}_t^k$ then
                    $\mathcal{G}_t^k.add(e)$;
    Send $\gamma \times$ ClientUpdate($k, w_i$) to central server in each FL iteration
```

Since the poisoning attack conducted by Xu et al [21] is on Euler+LANL, we focus on attacking the Euler GNIDS. We test both LANL and OpTC datasets to evaluate the generality of the attack. In Table V, we show the attack result when Euler is trained on LANL under ENTENTE. We select client 4 as the malicious client, which observes 495 malicious edges (out of 517 total malicious edges). Among these edges, 492 are cross-client edges, so the attacker has the motivation to poison the global model to hide their attack from the other clients. We define "success rate" (SR) as the ratio of malicious edges that evade detection, which is similar to Xu et al. [21]. We also count the number of injected edges per malicious edge (EPM). EPM could be larger than 1 when multiple snapshots are poisoned. γ is set to values between 5 and 100 and p is set to values between 25% to 100%. The row with "-" in γ and p in Table V shows result without attack. The attack is more powerful with larger γ and p. The result shows ENTENTE can bound SR to a low number (9.30%)[6]. When disabling the norm bounding defense (ENTENTE-UB), the training GNIDS model observes higher SR, but more importantly, "NaN" (not a number) error occurs with larger p and γ. In this case, the gradient updates accumulate substantial changes, leading to sudden large adjustments of the global model weights and gradient explosion. Training the GNIDS model will fail, which maps to the untargeted attack against FL [73].

Next, we change the malicious client from client 4 to client 1, 2, and 3 separately, and evaluate the impact. We also consider the setting of colluding clients and evaluate ENTENTE when both client 1 and 4 are malicious. We fix the attack parameters that γ=100, p=100%, and "EPM" to 80 to maximize attacker's capabilities. The result on LANL is shown in Table VI. SR is still low and we found client collusion does not lead to higher SR against ENTENTE.

[6]The implementation of Xu et al. sets the classification threshold of Euler manually and then injects edges according to the threshold. This setting differs from Euler's default setting which learns the classification threshold from a validation set. We follow Euler's default setting, which might lead to worse performance than Xu et al.

TABLE V
ATTACK ON LANL ($K = 4$). CLIENT 4 IS MALICIOUS. THE UPPER ("E") AND LOWER PARTS ("EUB") OF THE TABLE SHOWS THE RESULT OF ENTENTE AND ENTENTE-UB.

	p	γ	AP(%)	AUC(%)	SR(%)	EPM
E	-	-	0.72	97.00	-	
	25%	5	0.54	96.96	9.30	22
	50%	5	0.56	96.97	9.30	40
	50%	25	0.56	96.97	9.30	40
	75%	25	0.60	96.99	9.30	62
	100%	100	0.55	96.96	9.30	80
EUB	25%	5	0.74	97.34	11.4	22
	50%	25	0.18	96.23	40.59	22
	100%	100	NaN	NaN	-	80

TABLE VI
DIFFERENT MALICIOUS CLIENT ON LANL.

	Client	AP (%)	AUC (%)	SR(%)
E	1	0.37	96.54	6.60
	2	0.53	96.93	10.48
	3	0.55	96.96	8.9
	1, 4	0.58	96.90	7.48

Regarding OpTC, we choose $K = 3$ and use client 3 as a malicious client. Similar as the result on LANL, ENTENTE is robust even under very large p and γ, as shown in Table VII. Without the norm-bounding defense, much higher SR is observed or the training process cannot finish.

TABLE VII
ATTACK ON OPTC ($K = 3$). CLIENT 3 IS MALICIOUS. THE UPPER ("E") AND LOWER ("EUB") PARTS OF THE TABLE SHOWS THE RESULT OF ENTENTE AND ENTENTE-UB.

	p	γ	AP(%)	AUC(%)	SR(%)	EPM
E	-	-	83.29	99.76	-	
	10%	2	83.70	99.72	17.65	148
	25%	5	83.48	99.76	14.61	374
	50%	10	85.69	99.78	13.38	764
	100%	100	84.91	99.75	14.20	1513
EUB	10%	2	72.94	99.30	28.24	148
	50%	5	NaN	NaN	-	764
	100%	25	NaN	NaN	-	1513

E. Ablation Study

Due to space limit, we provide the results of ablation study in the extended version [22]. The results include impact of ACS, graph augmentation, clients' weight initialization, K and m, comparison between local and global models, and visualization of MBM clustering.

VI. DISCUSSION

Limitations and future works. First, we admit that the results of ENTENTE on LANL are far from ideal when the redteam events are TP. A few reasons have been given in prior works [4], [3], including: 1) the labeled malicious events are too coarse-grained; 2) the malicious activities after the redteam's ground-truth events are not tracked; 3) potentially malicious events on included in the "attack-free" training period. Second, due to the lack of ground truth of the locations of each device in the LANL or OpTC dataset, we choose to run the clustering method MBM [64] to generate FL clients' data. The location information is usually not provided from a log dataset and the IP is also anonymized. Clustering is our best effort. Similar approaches have been followed by other FGL works like [14], [35], [74]. Third, we did not simulate ENTENTE and other baseline FL methods in a fully distributed environment, i.e., different clients on different machines. Hence, the actual overhead, especially network communication, should be higher. However, the extra overhead caused by ENTENTE over the basic FedAVG method is introduced during bootstrap and ACS, and Section V-B shows it is reasonable. Finally, we did not simulate different FL poisoning attacks (surveyed in Section VII) besides Xu et al. [21], due to the gap between the attack methods and the concrete GNIDS setting. We leave the exploration of new attacks as future work.

Privacy implications of ENTENTE. Due to the usage of FL, the privacy attacks against FL, like membership inference attack (MIA) [75] and gradient inversion attack (GIA) [76], can be utilized to attack ENTENTE. To safeguard the privacy of FL clients, differential privacy (DP) can be applied to add noise to the model updates, which hides the existence of a single instance under FL. As such, DP directly defends against MIA [77]. Hatamizadeh et al. evaluated DP against GIA, and the results show DP-SGD is particularly effective in reducing the leakage from gradients [78]. Besides, poisoning attacks can be deterred by DP, as confirmed by previous studies [15], [77], [79], [16]. We have conducted a preliminary analysis regarding the combination of DP and ENTENTE. As detailed in Appendix VIII-E, our experimental results suggest that achieving strong defense against poisoning while preserving acceptable model utility remains a significant challenge.

Other options for cross-silo GNIDS. Cryptographic tools such as multiparty computation (MPC) and homomorphic encryption (HE) can be used to train a model using data from different regions with strong privacy protection. However, MPC and HE would incur much higher communication and computational overheads than FL, making them unsuitable to train GNIDS on large-scale log data now. CoGNN [80], the SOTA approach for MPC-based graph learning, communicates from 1GB to 4GB *per epoch* on small-scale datasets like Cora (2,708 nodes and 10,556 edges). The overhead will be amplified under larger GNIDS datasets. Meanwhile, ENTENTE transmits 1.94MB model weights in total when training Euler on LANL, as described in Section V-C. Moreover, CoGNN only supports static GNN models, but the SOTA GNIDS explicitly leverage temporal information with non-GNN models like GRU (integrated by Euler) and TGN (integrated by Jbeil). As a result, we believe FL is the most suitable approach for our problem setting at the current stage.

VII. RELATED WORK

Host-based Intrusion Detection (HIDS). This work focuses on developing privacy-preserving GNIDS, which consume network logs. Graph learning has also been applied to host logs. We refer interested readers to a survey by Inam et al. [24] for details. Below we provide a brief overview. The main approach to detect intrusions under HIDS is through *provenance tracking* [81], which performs variations of breadth-first search (e.g., under temporal constraints like happens-before relation [82]) to find other attack-related nodes given Indicators of Compromise (IoCs). Yet, provenance tracking often leads to a large number of candidate nodes to be investigated [83], and many HIDS add heuristics to reduce the investigation scope [84], [85]. Another way to reduce the false positives is to simplify the graphs, e.g., through event abstraction [86], [87] and graph summation [88].

Recently, like GNIDS, GNN has also been tested for HIDS and various embedding techniques have been developed/used, including masked representation learning [89], TGN [90], Graph2Vec [91], embedding recycling [92], root cause embedding [93], etc. It is likely that FL could benefit these HIDS, but a different operational model and/or learning method is needed. For instance, cross-device FL instead of cross-silo FL is a more suitable FL setup.

FL for security. Before our work, FL has been applied in various security-related settings, like risk modeling on mobile devices [94], malicious URL detection [95], detecting abnormal IoT devices [96], malware detection [97], browser fingerprinting detection [98], etc. Though FL has been used to train a model to detect intrusions with system logs collected from cloud instances [99] and security events collected from participating organizations [100], the models trained by these works (i.e., Transformer and RNN) are tailored to the tabular representation from logs. So far, none of the existing works have trained a GNN model with FL to detect intrusions from large-scale network logs.

Security and privacy for FL. Here, we present a more detailed overview of security and privacy issues in FL. Regarding security, poisoning (or backdoor) attacks are considered as the major threat [101], and the previous attacks can be divided into data poisoning [102], [103], [104] that manipulates the clients' training data and model poisoning [105], [106] that manipulates the training process or models themselves. In either case, poisoning will result in deviation of model updates, and the majority of defenses aim to detect and mitigate abnormal deviation [107], [108], [109], [110], [111]. ENTENTE integrates norm bounding [15] into the training procedure and the empirical and theoretical results show its effectiveness.

To defend FL against inference attacks [75], [76], differential privacy (DP) has been applied to FL. The noises can be added to each training step with DP-SGD [112], to the trained local model [113] and to the central server [114]. Recently, Yang et al. studied the accuracy degradation caused by FL+DP, and exploited client heterogeneity to improve the FL model's accuracy [115]. We attempted to integrate DP into ENTENTE but the accuracy drop is prominent. We also believe the privacy threat and the privacy notions need to be clearly defined under the GNIDS setting, in order to enable more effective defense.

VIII. Conclusion

In this paper, we propose ENTENTE, a new Graph-based Network Intrusion Detection Systems (GNIDS) backed by Federated Learning (FL), to address concerns in data sharing. We carefully tailor the design of ENTENTE to the unique settings of the security datasets (e.g., highly imbalanced classes and non-IID client data) and variations of GNIDS tasks/architectures. With new techniques like weight initialization and adaptive contribution scaling, we are able to achieve the desired design goals (effectiveness, scalability and robustness) altogether. The evaluation of three large-scale log datasets, namely OpTC, LANL and Pivoting, shows that ENTENTE outperforms the other baseline systems, even including the model trained on the whole dataset in some cases. We also conduct adaptive attacks against ENTENTE with FL poisoning, and show that ENTENTE can bound the attack success rate and ensure the training procedure can finish as desired. With ENTENTE, we hope to encourage more research to address the data sharing issues in building GNIDS, which is understudied, and develop better attack and defense benchmarks to evaluate the robustness of federated graph learning (FGL) systems.

Acknowledgment

We thank reviewers for the valuable suggestions. This work is supported by NSF CNS-2220434 and Amazon Research Awards. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the sponsors.

References

[1] Lockheed Martin, "Cyber kill chain," https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html, 2019.

[2] Z. Li and A. Oprea, "Operational security log analytics for enterprise breach detection," in *2016 IEEE Cybersecurity Development (SecDev)*. IEEE, 2016, pp. 15–22.

[3] J. Xu, X. Shu, and Z. Li, "Understanding and bridging the gap between unsupervised network representation learning and security analytics," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 12–12.

[4] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Transactions on Privacy and Security*, vol. 26, no. 3, pp. 1–36, 2023.

[5] J. Khoury, D. Klisura, H. Zanddizari, G. D. L. T. Parra, P. Najafirad, and E. Bou-Harb, "Jbeil: Temporal graph-based inductive learning to infer lateral movement in evolving enterprise networks," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 9–9.

[6] M. Rabbani, L. Rashidi, and A. A. Ghorbani, "A graph learning-based approach for lateral movement detection," *IEEE Transactions on Network and Service Management*, 2024.

[7] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[8] B. Imana, A. Korolova, and J. Heidemann, "Institutional privacy risks in sharing dns data," in *Proceedings of the Applied Networking Research Workshop*, 2021, pp. 69–75.

[9] F. Menges, T. Latzo, M. Vielberth, S. Sobola, H. C. Pöhls, B. Taubmann, J. Köstler, A. Puchta, F. Freiling, H. P. Reiser *et al.*, "Towards gdpr-compliant data processing in modern siem systems," *Computers & Security*, vol. 103, p. 102165, 2021.

[10] C. K. Lim, "Singapore - data protection overview," https://www.dataguidance.com/notes/singapore-data-protection-overview, 2023.

[11] Palo Alto Networks, "Strata logging service," https://docs.paloaltonetworks.com/strata-logging-service/administration/overview/supported-regions, 2025.

[12] Microsoft, "Data security and retention in microsoft defender xdr," https://learn.microsoft.com/en-us/defender-xdr/data-privacy, 2025.

[13] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[14] Y. Yao, W. Jin, S. Ravi, and C. Joe-Wong, "Fedgcn: Convergence and communication tradeoffs in federated training of graph convolutional networks," *arXiv preprint arXiv:2201.12433*, 2022.

[15] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[16] X. Zhang, X. Chen, M. Hong, S. Wu, and J. Yi, "Understanding clipping for federated learning: Convergence and client-level differential privacy," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 26 048–26 067.

[17] Five Directions, "Operationally transparent cyber (optc) data release," https://github.com/FiveDirections/OpTC-data, 2020.

[18] A. D. Kent, "Comprehensive, Multi-Source Cyber-Security Events," Los Alamos National Laboratory, 2015.

[19] G. Apruzzese, F. Pierazzi, M. Colajanni, and M. Marchetti, "Detection and threat prioritization of pivoting attacks in large networks," *IEEE transactions on emerging topics in computing*, vol. 8, no. 2, pp. 404–415, 2017.

[20] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.

[21] X. Xu, Q. Hao, Z. Yang, B. Li, D. Liebovitz, G. Wang, and C. Gunter, "How to cover up anomalous accesses to electronic health records," in *32nd {USENIX} Security Symposium ({USENIX} Security 23)*, 2023.

[22] J. Xu, C. Li, Y. Zheng, and Z. Li, "Entente: Cross-silo intrusion detection on network log graphs with federated learning," *arXiv preprint arXiv:2503.14284*, 2025.

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

[24] M. A. Inam, Y. Chen, A. Goyal, J. Liu, J. Mink, N. Michael, S. Gaur, A. Bates, and W. U. Hassan, "Sok: History is a vast early warning system: Auditing the provenance of system intrusions," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2620–2638.

[25] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[27] C. Huang, J. Huang, and X. Liu, "Cross-silo federated learning: Challenges and opportunities," *arXiv preprint arXiv:2206.12949*, 2022.

[28] R. Liu, P. Xing, Z. Deng, A. Li, C. Guan, and H. Yu, "Federated graph neural networks: Overview, techniques and challenges," *arXiv preprint arXiv:2202.07256*, 2022.

[29] X. Fu, B. Zhang, Y. Dong, C. Chen, and J. Li, "Federated graph machine learning: A survey of concepts, techniques, and applications," *ACM SIGKDD Explorations Newsletter*, vol. 24, no. 2, pp. 32–47, 2022.

[30] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *International Conference on Learning Representations*, 2020.

[31] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[32] X. Li, Z. Wu, W. Zhang, Y. Zhu, R.-H. Li, and G. Wang, "Fedgta: Topology-aware averaging for federated graph learning," *Proceedings of the VLDB Endowment*, vol. 17, no. 1, pp. 41–50, 2023.

[33] Y. Wang, S. Guo, D. Qiao, G. Liu, and M. Li, "Fedsg: A personalized subgraph federated learning framework on multiple non-iid graphs," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024.

[34] C. Chen, W. Hu, Z. Xu, and Z. Zheng, "Fedgl: Federated graph learning framework with global self-supervision," *arXiv preprint arXiv:2105.03170*, 2021.

[35] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6671–6682, 2021.

[36] Y. Zhu, X. Li, Z. Wu, D. Wu, M. Hu, and R.-H. Li, "Fedtad: Topology-aware data-free knowledge distillation for subgraph federated learning," *arXiv preprint arXiv:2404.14061*, 2024.

[37] Z. Zhang, M. Chen, M. Backes, Y. Shen, and Y. Zhang, "Inference attacks against graph neural networks," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4543–4560.

[38] IBM, "What is SIEM?" https://www.ibm.com/topics/siem, 2023.

[39] E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22), USENIX Association, Boston, MA*, 2022.

[40] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2019.

[41] S. Dambra, L. Bilge, and D. Balzarotti, "A comparison of systemic and systematic risks of malware encounters in consumer and enterprise environments," *ACM Transactions on Privacy and Security*, vol. 26, no. 2, pp. 1–30, 2023.

[42] A. Bates, D. J. Tian, K. R. Butler, and T. Moyer, "Trustworthy {Whole-System} provenance for the linux kernel," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 319–334.

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

[44] V. Gandhi, S. Banerjee, A. Agrawal, A. Ahmad, S. Lee, and M. Peinado, "Rethinking system audit architectures for high event coverage and synchronous log availability," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 391–408.

[45] Microsoft, "Get-aduser," https://learn.microsoft.com/en-us/powershell/module/activedirectory/get-aduser?view=windowsserver2022-ps, 2023.

[46] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "Shadewatcher: Recommendation-guided cyber threat analysis using system audit records," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 489–506.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[49] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[50] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[51] C. Meng, S. Rambhatla, and Y. Liu, "Cross-node federated graph neural network for spatio-temporal data modeling," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 1202–1211.

[52] Z. Wang, W. Kuang, Y. Xie, L. Yao, Y. Li, B. Ding, and J. Zhou, "Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4110–4120.

[53] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

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

[55] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[56] M. Drobyshevskiy and D. Turdakov, "Random graph modeling: A survey of the concepts," *ACM computing surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[57] H. Bunke, "On a relation between graph edit distance and maximum common subgraph," *Pattern recognition letters*, vol. 18, no. 8, pp. 689–694, 1997.

[58] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[59] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1078–1088, 2021.

[60] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *ISOC Network and Distributed System Security Symposium (NDSS)*, 2021.

[61] V. Siomos and J. Passerat-Palmbach, "Contribution evaluation in federated learning: Examining current approaches," *arXiv preprint arXiv:2311.09856*, 2023.

[62] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[63] R. Paudel and H. H. Huang, "Pikachu: Temporal walk based dynamic graph embedding for network anomaly detection," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–7.

[64] C. Larroche, "Multilayer block models for exploratory analysis of computer event logs," in *International Conference on Complex Networks and Their Applications*. Springer, 2022, pp. 625–637.

[65] ——, "Multilayerblockmodels," https://github.com/cl-anssi/MultilayerBlockModels, 2022.

[66] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[67] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Pedestrian detection with spatially pooled features and structured ensemble learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 6, pp. 1243–1257, 2015.

[68] iHeartGraph, "Euler source code," https://github.com/iHeartGraph/Euler, 2022.

[69] LMscope, "Jbeil source code," https://github.com/LMscope/Jbeil, 2024.

[70] S. Yan, Z. Li, C. Wu, M. Pang, Y. Lu, Y. Yan, and H. Wang, "You are your own best teacher: Achieving centralized-level performance in federated learning under heterogeneous and long-tailed data," *arXiv preprint arXiv:2503.06916*, 2025.

[71] A. Goyal, X. Han, G. Wang, and A. Bates, "Sometimes, you aren't what you do: Mimicry attacks against provenance graph host intrusion detection systems," in *NDSS*, 2023.

[72] K. Mukherjee, J. Wiedemeier, T. Wang, J. Wei, F. Chen, M. Kim, M. Kantarcioglu, and K. Jee, "Evading {Provenance-Based}{ML} detectors with adversarial system actions," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1199–1216.

[73] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[74] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-iid graphs," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 839–18 852, 2021.

[75] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[76] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 2512–2520.

[77] C. Xie, Y. Long, P.-Y. Chen, Q. Li, S. Koyejo, and B. Li, "Unraveling the connections between privacy and certified robustness in federated learning against poisoning attacks," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1511–1525.

[78] A. Hatamizadeh, H. Yin, P. Molchanov, A. Myronenko, W. Li, P. Dogra, A. Feng, M. G. Flores, J. Kautz, D. Xu *et al.*, "Do gradient inversion attacks make federated learning unsafe?" *IEEE Transactions on Medical Imaging*, vol. 42, no. 7, pp. 2044–2056, 2023.

[79] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[80] Z. Zou, Z. Liu, J. Shan, Q. Li, K. Xu, and M. Xu, "CoGNN: Towards secure and efficient collaborative graph learning," in *CCS*, 2024.

[81] X. Jiang, A. Walters, D. Xu, E. H. Spafford, F. Buchholz, and Y.-M. Wang, "Provenance-aware tracing ofworm break-in and contaminations: A process coloring approach," in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*. IEEE, 2006, pp. 38–38.

[82] X. Shu, F. Araujo, D. L. Schales, M. P. Stoecklin, J. Jang, H. Huang, and J. R. Rao, "Threat intelligence computing," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1883–1898.

[83] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, and G. Jiang, "High fidelity data reduction for big data security dependency analyses," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 504–516.

[84] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1137–1152.

[85] M. N. Hossain, S. Sheikhi, and R. Sekar, "Combating dependence explosion in forensic analysis using alternative tag propagation semantics," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1139–1155.

[86] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2020.

[87] L. Yu, S. Ma, Z. Zhang, G. Tao, X. Zhang, D. Xu, V. E. Urias, H. W. Lin, G. Ciocarlie, V. Yegneswaran *et al.*, "Alchemist: Fusing application and audit logs for precise attack provenance without instrumentation," in *NDSS*, 2021.

[88] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "Depcomm: Graph summarization on system audit logs for attack investigation," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 540–557.

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

[90] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "Kairos: Practical intrusion detection and investigation using whole-system provenance," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 5–5.

[91] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "Prographer: An anomaly detection system based on provenance graph embedding," in *32nd {USENIX} Security Symposium ({USENIX} Security 23)*, 2023.

[92] M. U. Rehman, H. Ahmadi, and W. U. Hassan, "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 139–139.

[93] A. Goyal, G. Wang, and A. Bates, "R-caid: Embedding root cause analysis within provenance-based intrusion detection," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 257–257.

[94] H. Fereidooni, A. Dmitrienko, P. Rieger, M. Miettinen, A.-R. Sadeghi, and F. Madlener, "Fedcri: Federated mobile cyber-risk intelligence," in *Network and Distributed Systems Security (NDSS) Symposium*, 2022.

[95] T. Ongun, S. Boboila, A. Oprea, T. Eliassi-Rad, J. Hiser, and J. Davidson, "Celest: Federated learning for globally coordinated threat detection," *arXiv preprint arXiv:2205.11459*, 2022.

[96] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[97] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

[98] M. S. M. S. Annamalai, I. Bilogrevic, and E. De Cristofaro, "Fp-fed: Privacy-preserving federated detection of browser fingerprinting," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2024.

[99] G. De La Torre Parra, L. Selvera, J. Khoury, H. Irizarry, E. Bou-Harb, and P. Rad, "Interpretable federated transformer log learning for cloud threat forensics," *NDSS 22*, 2022.

[100] M. Naseri, Y. Han, E. Mariconti, Y. Shen, G. Stringhini, and E. De Cristofaro, "Cerberus: exploring federated prediction of security events," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2337–2351.

[101] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[102] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur.(DISS)*, 2020, pp. 1–7.

[103] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 508–519.

[104] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International conference on learning representations*, 2019.

[105] H. Li, Q. Ye, H. Hu, J. Li, L. Wang, C. Fang, and J. Shi, "3dfed: Adaptive and extensible framework for covert backdoor attack in federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1893–1907.

[106] T. Krauß, J. König, A. Dmitrienko, and C. Kanzow, "Automatic adversarial adaption for stealthy poisoning attacks in federated learning," in *To appear soon at the Network and Distributed System Security Symposium (NDSS)*, 2024.

[107] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," *arXiv preprint arXiv:2201.00763*, 2022.

[108] K. Kumari, P. Rieger, H. Fereidooni, M. Jadliwala, and A.-R. Sadeghi, "Baybfed: Bayesian backdoor defense for federated learning," *arXiv preprint arXiv:2301.09508*, 2023.

[109] H. Fereidooni, A. Pegoraro, P. Rieger, A. Dmitrienko, and A.-R. Sadeghi, "Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning," in *NDSS*, 2024.

[110] P. Rieger, T. Krauß, M. Miettinen, A. Dmitrienko, and A.-R. Sadeghi, "Crowdguard: Federated backdoor detection in federated learning," in *To appear soon at the Network and Distributed System Security Symposium (NDSS)*, 2024.

[111] E. Kabir, Z. Song, M. R. U. Rashid, and S. Mehnaz, "Flshield: a validation based federated learning framework to defend against poisoning attacks," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2572–2590.

[112] L. Sun, J. Qian, and X. Chen, "Ldp-fl: Practical private aggregation in federated learning with local differential privacy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[113] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5201–5212.

[114] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations*, 2018.

[115] Y. Yang, B. Hui, H. Yuan, N. Gong, and Y. Cao, "{PrivateFL}: Accurate, differentially private federated learning via personalized data transformation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1595–1612.

[116] DARPA, "Transparent computing (archived)," https://www.darpa.mil/program/transparent-computing, 2014.

[117] Microsoft, "NTLM user authentication," https://docs.microsoft.com/en-us/troubleshoot/windows-server/windows-security/ntlm-user-authentication, 2021.

Appendix

A. Algorithm of BA Model

In Algorithm 3, we describe BA model with pseudo-code.

B. Workflow of Entente

In Algorithm 4, we summarize the workflow of Entente. Specifically, w_i contains the model parameters for the graph encoder, temporal encoder and decoder, which are denoted by ENC$(\cdot)$, TEMP$(\cdot)$ and DEC$(\cdot)$. For the decoder, only the trainable implementations go through FL. JS$(\cdot)$ is Jaccard

Algorithm 3: Barabási–Albert (BA) Model

```
Data: $n$ (number of nodes), $m$ (number of edges to
      attach from a new node to existing nodes)
Result: $G$ (generated graph)
$G \leftarrow$ InitializeGraph($m$);
for $i = m + 1$ to $n$ do
    AddNode($G, i$);
    $k_j \leftarrow$ Degree($G$);
    $K \leftarrow \sum_{j \in G} k_j$;
    $L \leftarrow$ EmptyList();
    foreach $j \in G$ do
        $L$.Add($j, k_j / K$);
    $S \leftarrow$ RandomSelect($L, m$);
    foreach $v \in S$ do
        AddEdge($G, i, v$);
return $G$
```

similarity. $\nabla \mathcal{L}$ computes gradients on the training loss, and the same loss function of the GNIDS is used.

C. Datasets and pre-processing

The OpTC dataset [17] contains the telemetry data collected under the DARPA TC program [116], during which APT attacks were simulated on different OSes. The host-level activities between subjects like processes and objects like files and sockets were logged. Like Euler [4], we use the "START" events related to the "FLOW" objects (i.e., network flows). The nodes are hosts distinguished by IP addresses and the flows between hosts are merged into edges. The statistics after this step are shown in Table II. We split the data into 6 minutes (360s) window. We use the first 5 days' snapshots (no redteam events exist) for training and the remaining 3 days' snapshots for testing. OpTC is only tested by Euler and we report its result accordingly.

The LANL dataset [18] contains anonymized event data from four sources within Los Alamos National Laboratory's internal computer network. We use the authentication logs from individual computers and domain controller servers following [4], [5]. Simulated redteam attacks are conducted from the compromised machines. As shown in Table II, the malicious events consist of a fairly small portion among all events, posing a great challenge to GNIDS. Regarding data pre-processing, like [4], [5], we only keep the events with the keyword "NTLM" [117], as other events are unrelated to authentications. For Euler, the logs are split by a 30-minute (1,800s) window into snapshots. We use "source computer" and "destination computer" as nodes and all events sharing the same pairs of nodes are merged into an edge. The first 42 hours are used to train the Euler GNIDS. On average, each snapshot has 7,957 edges and we use 5% edges for validation. After 42 hours, redteam events appear and the following snapshots are used for testing. For Jbeil, we follow their pre-processing procedures to use the events that happened during the first

Algorithm 4: Global model training under ENTENTE.

```
Data: $E$ is the number of local epochs; $\eta$ is the
      learning rate; Number of maximum FL
      iterations $R$; Number of subgraphs $T$; $c_1$ and $c_2$
      are two constants.
Result: Global model parameters $w_{i+1}$
Server executes:
Initialize $w_1$;
$\mathcal{G}^{ref} \leftarrow$ BA_Model($n$, $m$);
foreach client k from all clients do
    $S^k_{Jac} \leftarrow$ ClientInitialWeight($k, \mathcal{G}^{ref}$);
for $i = 1$ to $R$ do
    foreach client k from all clients do
        $w^k_i \leftarrow$ ClientUpdate($k, w_i$);
    $S^K_i, D^K_i =$ ACS($w_i, w^k_i$);
    $w_{i+1}$[ENC] $\leftarrow w_i$[ENC] + $\Sigma^K_{k=1}(c_1 \times S^k_{Jac} + c_2 \times$
        $S^k_i \times D^k_i) \times$ NB($\Delta w^k_{i+1}$[ENC]) ;
    $w_{i+1}$[TEMP] $\leftarrow w_i$[TEMP] $+ \Sigma^K_{k=1}(c_1 \times S^k_{Jac} +$
        $c_2 \times S^k_i \times D^k_i) \times$ NB($\Delta w^k_{i+1}$[TEMP] ;
    if DEC is trainable then
        $w_{i+1}$[DEC] $\leftarrow w_i$[DEC] $+ \Sigma^K_{k=1}(c_1 \times S^k_{Jac} +$
            $c_2 \times S^k_i \times D^k_i) \times$ NB($\Delta w^k_{i+1}$[DEC] ;
    if early_stopping($w_1, \ldots, w_{i+1}$) then
        break;
return $w_{i+1}$
Function ClientInitialWeight($k, \mathcal{G}^{ref}$):
    Generate local graph $\mathcal{G}^k$;
    Add cross-client edges to $\mathcal{G}^k$;
    $[\mathcal{G}^k_1, \ldots, \mathcal{G}^k_T] \leftarrow$ separate($\mathcal{G}^k$);
    return JS(WLH($\mathcal{G}^k$), WLH($\mathcal{G}^{ref}$)) to server;
Function ACS($w_i, w^k_i$):
    Compute $S^K_i$ and $D^K_i$ ;
    return $S^K_i, D^K_i$ to server;
Function ClientUpdate($k, w$):
    for each local epoch i from 1 to E do
        $w \leftarrow w - \eta \nabla \mathcal{L}(w; [\mathcal{G}^k_1, \ldots, \mathcal{G}^k_T])$;
    return $w$ to server;
```

14 days, including 12,049,423 events. Different from Euler, Jbeil did not use the LANL's redteam events as malicious samples. Instead, it injects non-existent edges (i.e., negative sampling) and conducts link prediction (i.e., predicting an edge's existence in testing)[7]. We use 70% events for training, 15% for validation and the remaining 15% for testing. For the transductive learning mode, the training and testing sets share the same set of nodes. For the inductive learning mode, 30% of the nodes are hidden during the training but unmasked

[7]Jbeil has another mode that uses a lateral-movement simulator to inject malicious edges, but this simulator has not been released. We have contacted the authors and confirmed it.

TABLE VIII
EVALUATION ON PIVOTING WITH JBEIL. NON-FL IS LISTED FOR EASE OF COMPARISON. ALL NUMBERS ARE USED IN THE PERCENT FORMAT. THE BEST AND THE SECOND-BEST AMONG FL METHODS ARE HIGHLIGHTED RESPECTIVELY. ⬆MEANS OUTPERFORMING NON-FL AND ▼MEANS THE WORST PERFORMANCE.

	Pivoting-Jbeil-Transductive						**Pivoting-Jbeil-Inductive**					
Client#	**2**		**3**		**4**		**2**		**3**		**4**	
Algorithm	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC
Non-FL	96.26	97.05	96.26	97.05	96.26	97.05	95.92	96.87	95.92	96.87	95.92	96.87
FedAVG	82.07	88.70	68.81▼	75.70▼	81.19	87.26	83.59	89.46	71.05▼	78.06▼	80.80	86.99
FedOpt	95.81	97.02	81.14	87.37	90.28	93.97	96.21⬆	97.44⬆	85.70	90.87	91.65	94.94
FedProx	81.01	87.34	80.72	86.71	58.62▼	62.56▼	80.58	87.17	81.53	87.40	58.08▼	61.99▼
FedAVG-N	74.14▼	80.64▼	93.38	96.21	95.79	97.50⬆	72.82▼	78.81▼	87.62	92.41	92.68	95.84
ENTENTE-UB	93.55	96.00	89.72	93.91	94.78	96.24	91.46	94.86	87.53	92.58	90.96	93.97
ENTENTE	95.54	97.21⬆	90.98	94.68	96.40⬆	97.77⬆	92.88	95.85	89.05	93.58	93.19	96.09

during testing, following Jbeil's setting.

The Pivoting dataset [19] comprises network flow data, collected over a full working day under a large organizational setting, exclusively representing internal-to-internal communications among hosts. The simulated pivoting activities for lateral movement are considered malicious. It is only tested by Jbeil and we follow its setting and use the first 1,000,000 events, which correspond to 698 nodes. Again, the data is split into 70% for training, 15% for validation, and the remaining 15% for testing. Link prediction is conducted and we test both transductive and inductive learning modes.

D. More Results on LANL, OpTC and Pivoting Dataset

Table VIII shows the results on the Pivoting dataset when Jbeil is the GNIDS. We observe that ENTENTE achieves the best AUC in most cases ($K = 2, 4$ for transductive mode and $K = 3, 4$ for inductive mode). FedOpt outperforms ENTENTE when $K = 2$ and ENTENTE-UB outperforms when $K = 3$ for transductive but the differences are small (less than 5%). Though non-FL achieves the best AP and AUC in nearly all cases, the margins over ENTENTE are also small (less than 7% AP and 4% AUC).

E. Differential Privacy for ENTENTE

Specifically, Equation 10 can be adjusted to integrate DP as written below:

$$w_{i+1} = w_i + \frac{1}{K}\sum_{k=1}^{K} r_k^k \times \mathsf{NB}(\Delta w_{i+1}^k) + \mathcal{N}(0, M_{\mathsf{qs}}^2\sigma^2) \quad (12)$$

We are able to support weak DP [15] and CDP [79], [16] defenses with Equation 12. For theoretical completeness, ENTENTE realizes the DP guarantee parameterized by $\epsilon = O(KM_{qs}\sqrt{R\log(1/\delta)}/\sigma)$.

Theorem 3. *Consider that K clients collaboratively train a model in* ENTENTE *for R rounds. Through Equation 12,* ENTENTE *realizes $(KM_{qs}\sqrt{R\log(1/\delta)}/\sigma, \delta)$-DP.*

Their difference is the value of M_{qs}. Weak DP extends norm bounding by adding a Gaussian noise under small variance σ to each model update, and the value of M_{qs} is relatively smaller than expected as the standard DP. CDP requires the noise level to be proportional to the sensitivity.

F. Review of Supporting Lemmas

Lemma 1 (Lipschitz Gradient)**.** *The function F_k is L-smooth for any $k \in [K]$ such that,*

$$\|\nabla F_k(x) - \nabla F_k(y)\| \le L\|x-y\|, \quad \forall\, x, y \in \mathbb{R}^d \quad (13)$$

Lemma 2 (Bounded Local Variance)**.** *The function F_k has σ_{local}-bounded local variance such that,*

$$\mathbb{E}[\|\nabla[f_k(w,z)]_j - [\nabla F_k(w)]_j\|] = \sigma^2_{\mathsf{local},j} \quad (14)$$

for all $w \in \mathbb{R}^d, j \in [d]$, and $k \in [K]$.

Lemma 3 (Bounded Global Variance)**.** *The function F_k has σ_{global}-bounded global variance such that,*

$$\frac{1}{K}\sum_{k=1}^{K}\|\nabla[F_k(w)]_j - [\nabla f(w)]_j\| \le \sigma^2_{\mathsf{global},j} \quad (15)$$

for all $w \in \mathbb{R}^d$ and $j \in [d]$.

Lemma 4 (Bounded Gradients)**.** *The gradients of function $f_k(w, z)$ is G-bounded such that,*

$$|[\nabla f_k(w,z)]_j \le G|, \quad \forall\, j \in [d] \quad (16)$$

for any $k \in [K], w \in \mathbb{R}^d$ and $z \sim \mathrm{data}_k$.

Due to space limit, all the proofs are elaborated in the extended version [22].

G. Artifact Appendix

The artifact evaluation process is designed to validate the repeatability and usability of the results presented in the research paper "ENTENTE: Cross-silo Intrusion Detection on Network Log Graphs with Federated Learning." The paper introduces ENTENTE, a novel federated learning framework designed for Graph-based Network Intrusion Detection Systems (GNIDS) for regulation compliance. The primary objective of ENTENTE is to achieve effectiveness, scalability and robustness simultaneously. The evaluation process comprises two main components: training and testing the IDS. As a valuable resource, the authors provide a GitHub link that grants access to the source code, data, and scripts necessary for reproducing the results described in the paper.

By offering these artifacts, the researchers enable fellow researchers and practitioners to replicate and build upon their work in Network Log detection. The artifacts include comprehensive software, data, and scripts employed to generate the findings presented in the paper. The accessibility of the GitHub repository ensures transparency. It fosters collaboration among researchers, facilitating advancements in the domain of Network Log detection and contributing to the overall improvement of security systems.

1) Description & Requirements: To support the reproduction of our results, we have provided the code, sample data, models, and intermediate files required to produce evasive attacks from the evaluation. Although our artifacts make no particular assumptions about computing power, 30GB of disk space and 32GB of memory are required to store and run the models, data samples, and software dependencies.

How to access Our code and data can be accessed at https://zenodo.org/records/16955361 and the DOI is https://doi.org/10.5281/zenodo.16955361.

Hardware dependencies Running our experiments requires approximately 30GB of disk storage to accommodate the data, codebase. To efficiently load and process the datasets and model weights during execution, a system with at least 32GB of system memory (RAM) is recommended. While the experiments can be run entirely on CPU—taking around 30 minutes per run—they can be significantly accelerated with GPU support. Specifically, using two NVIDIA RTX 3090 GPUs (each with 32GB of VRAM) can reduce the runtime to approximately 5 minutes, depending on the specific task.

Software dependencies Our code is written in Python and uses Miniconda for environment management. Miniconda can be installed from https://conda.io/projects/conda/en/latest/user-guide/install/index.html. Simply follow the installation directions for your machine architecture. Python 3.9 will be installed as part of the environment building process. Our experiments were run on Ubuntu 22.

Benchmarks All the datasets and models required for use with this artifact are provided in the Zenodo record.

2) Artifact Installation & Configuration: Commands are listed below.

```
# GPU environment
conda create -n entente python==3.9 -y
conda activate entente
pip install torch==1.10.1+cu111 torchvision==0.11.2+
    cu111 -f https://download.pytorch.org/whl/cu111/
    torch_stable.html
pip install -r requirements.txt
pip install torch_scatter torch_sparse torch_cluster
     torch_spline_conv -f https://data.pyg.org/whl/
    torch-1.10.1+cu111.html --no-index
```

```
# CPU environment
conda create -n entente python==3.9 -y
conda activate entente
pip install torch==1.10.1 torchvision==0.11.2
pip install -r requirements.txt
pip install torch_scatter torch_sparse torch_cluster
     torch_spline_conv -f https://data.pyg.org/whl/
    torch-1.10.1+cpu.html --no-index
```

3) Experiment Workflow: We provide detailed instructions on how to run the system. In the README, we also provide the exact commands to run for each major component.

4) Evaluation:

- Download Dataset and Code;
- Run the code for the Euler system on the OpTC dataset.

```
    python Euler/run.py --cluster_fname optc_2_2
    .json --client_number 3 --epochs 1
    python Euler/run.py --cluster_fname optc_3_3
    .json --client_number 4 --epochs 1

```

- Run the code for the Jbeil system on the LANL dataset.

```
    python Jbeil/run.py --client_number 3
    python Jbeil/run.py --client_number 4

```