FL-TFlow: Benign-Only Federated LoRA Tuning of SLMs for Edge Ransomware Detection

Wallace P. Cruz[1], Jose Pinto[1], Thiago Marques[1], Rafael Veiga[1], Hugo Santos[41], Lucas Bastos[2], Gabriel Talasso[5], Allan Costa[3], Denis Rosário[1], and Eduardo Cerqueira [1]

[1]Federal University of Pará (UFPA)

[2]Federal University of South and Southeast of Pará (UNIFESPA)

[3]Federal Rural University of the Amazon (UFRA)

[4]State University of Pará (UEPA)

[5]University of Campinas (UNICAMP)

{wallace.cruz,jose.pinto}@itec.ufpa.br, hugo.santos@uepa.br

{thiago.marques,rafael.teixeira.silva}@icen.ufpa.br,

bastos.lucas@unifesspa.edu.br g235078@dac.unicamp.br,

allan.costa@ufra.edu.br,{denis,cerqueira}@ufpa.br

Abstract. *Ransomware is a major problem for edge computing environments because traditional centralized detection methods do not work as well when resources are limited, and privacy is a concern. Federated Learning (FL) addresses privacy concerns by enabling people to train models together without sharing raw data. Small Language Models (SLMs) are good for devices with limited resources because they are easy to use. In this paper, we introduce FL-TFlow, a benign-only detection framework based on textual representations of network flows that uses Low-Rank Adaptation (LoRA) to fine-tune SLMs in a federated setting. By training exclusively on benign traffic, the model detects attacks through degradation in token-prediction scores derived from the SLM output, eliminating the need for labeled attack data. Our evaluation results reveal that FL-TFlow achieves a flow-level F1-score of 0.9379 in the benign-only scenario. In addition, FL-TFlow reduces the false positive rate (FPR) by approximately 0.01% compared to the baseline schemes, i.e., reducing false alarms.*

1. Introduction

The Internet of Things (IoT) is growing very quickly. By 2030, there could be as many as 30 billion connected devices [Vailshery 2024]. IoT devices are increasingly monitoring critical infrastructure, from industrial sensors to healthcare systems. In this way, IoT becomes a prime target for advanced cyber threats, especially ransomware. In these high-stakes situations, the effects go beyond data encryption and can include severe service outages. However, traditional Intrusion Detection Systems (IDS) rely on fixed signatures, which are increasingly ineffective against these polymorphic threats [Buyuktanir et al. 2025]. Hence, it is important to adopt a paradigm shift toward adaptive, data-driven defense mechanisms.

Transformers, as a layer-based model, are widely used for language modeling by tokenizing sentences to capture their semantic structure. Using the tokens for semantic sentences, the layer focuses on learning patterns through self-supervised masking tasks [Guo et al. 2021]. Other methods, such as transfer learning, are often used in distributed training to classify malware logs and system call sequences [Sánchez et al. 2024]. Therefore, transformers learn the grammar of digital environments, identifying deviations as potential threats without relying on rigid signatures.

Despite these advances, deploying such powerful models in distributed IoT environments presents three challenges: privacy, bandwidth, and computational constraints. That is because transmitting raw telemetry from IoT devices to a central server violates privacy principles and saturates the network bandwidth. In this sense, Federated Learning (FL) mitigates these issues by enabling decentralized training through the aggregation of local updates without sharing raw data [McMahan et al. 2017]. However, standard FL introduces a communication bottleneck when applied to Large Language Models (LLMs), as transmitting billions of parameters across edge networks is prohibitively expensive.

Applications of LLMs in cybersecurity rely on system telemetry, such as logs, system calls, and network flows, which possess a grammar analogous to natural language. However, network flow logs are still non-semantic, numerical, and encrypted, posing unique challenges for tokenization and modeling across neural network architectures. This supervised paradigm often involves an unbalanced, unlabeled dataset containing both benign and malicious instances, which makes generalization during training difficult. Robustness learning with LLMs still depends on abundant data and the types of training and expected responses, which can identify significant deviations indicative of novel or zero-day ransomware attacks without relying on examples of every attack variant during training [Alsuwaiket 2025, Popoola et al. 2021].

To have a more specific model, one of the most common approaches is to use Small Language Models (SLMs), which can be more direct in making certain functions to improve inference times. Recent literature addresses the communication bottleneck with Parameter-Efficient Fine-Tuning (PEFT). Some works use FL scenarios to apply anomaly logs to workflow detection, using pre-trained SLMs like Low-Rank Adaptation (LoRA) for fine-tuning trainable parameters to reduce generalization communication while maintaining accuracy [Talasso et al. 2025]. However, this approach leverages the inherent semantic richness of textual logs. In addition, it is challenging to apply this approach to network flows composed of numerical, encrypted, and non-semantic data. Furthermore, existing IoT-specific LLM approaches rely on supervised learning with labeled attack data [Ferrag et al. 2024]. Although accurate for known threats, these models are fundamentally blind to zero-day ransomware campaigns that lack prior labels. Consequently, to the best of our knowledge, there is currently no framework that combines communication efficiency, the flow-based applicability of traditional IDS, and a benign-only training strategy capable of detecting unknown ransomware threats.

This paper introduces FL-TFlow, a FL framework designed for the benign-only fine-tuning of SLM specifically for ransomware detection in IoT/Edge network flows. FL-TFlow uses the benign-only training method, meaning it trains only on benign traffic to detect ransomware by looking for a drop in token-prediction scores. That makes the training method for zero-day attacks more efficient by classifying a novel ransomware log

that the model has not seen before. In its operation, FL-TFlow integrates LoRA into an FL framework to adapt the models to the essential gradients, compressing the model and making it more specific to the data each client is training on, ensuring that our system is both privacy-preserving and communication-efficient. Our evaluation results show that FL-TFlow achieves a flow-level F1-score of 0.9379, while FedMLLM reaches only 0.0264. Also, FL-TFlow reduces the benign false-positive rate (FPR) to 0.0122%, compared to 0.6674% for FedMLLM, resulting in substantially fewer false alarms. In addition, at the 30s window level, FL-TFlow achieves an F1-score of 0.8111, whereas FedMLLM reaches only 0.0276. The main contributions of this paper can be summarized as follows: i) a leakage-controlled Flow-to-Text serialization method that enables anomaly detection in non-semantic network traffic; ii) a federated LoRA architecture for network flows in the distinct domain of encrypted ransomware traffic; iii) a benign-only training strategy to detect unknown ransomware strains, validated through extensive experiments on realistic IoT datasets.

We organize the remainder of this paper as follows. Section 2 reviews the related work. Section 3 details the FL-TFlow framework. Section 4 presents the experimental setup and evaluation results. Finally, Section 5 concludes the paper.

2. Related Work

Guo *et al.* introduce LogBERT, which trains on log sequences with self-supervised objectives to detect anomalies as departures from learned behavior. Transformer-based log-anomalous detection has shown that language-modeling objectives can capture standard patterns and flag deviations. However, that approach does not account for the similarity of text in non-parser models or the use of nonpretrained models, which can limit the model's understanding during initial training [Guo et al. 2021].

Chen *et al.* developed BERT-Log, which uses a pre-trained language model and fine-tunes it for log anomaly detection, reporting strong performance on standard log datasets. Despite their effectiveness for log telemetry, these approaches typically assume centralized training and often depend on dataset-specific parsing or template extraction, which can limit portability and complicate deployment across heterogeneous environments. However, that paper does not consider the differences between the temporal use of SLM for classification [Chen and Liao 2022].

Talasso *et al.* demonstrate that combining FL with LoRA adapters enables benign-only anomaly detection and drastically reduces communication, as clients exchange only lightweight adapters. This line provides strong motivation for FL and PEFT in security telemetry. However, it is primarily validated using system logs. It does not explicitly study non-IID client heterogeneity or deployment-oriented temporal metrics (e.g., time-to-detection), which are critical when ransomware damage increases with detection delay [Talasso et al. 2025].

Maasaoui *et al.* transform flow records into feature value tokens, train a byte-level BPE tokenizer, and fine-tune BERT for multi-class intrusion detection on datasets including Edge-IIoT. For network-flow IDS, this paper shows that "tabular-to-text" encodings allow transformer models to operate in network traffic. These results support the feasibility of representing network flows as sequences. However, that approach does not account for the formulation's supervision and centralization, making it neither directly compara-

ble to benign-only novelty detection nor representative of privacy-preserving multi-client deployments [Maasaoui et al. 2024].

Sanchez *et al.* explores transfer learning with pre-trained language models in a long context for malware detection from system call sequences, highlighting operational trade-offs such as context size and the role of decision thresholds. Although the data type differs from that used for ransomware detection in network flows, the setting is not federated. This paper reinforces the need for training across a variety of client environments and data, rather than relying on a single accuracy metric, while accounting for thresholding and operational constraints [Sánchez et al. 2024].

Table 1 provides a comprehensive overview of the key attributes of the reviewed studies on training SLMs with adaptive mechanisms to preserve privacy and improve training efficiency for ransomware detection. Based on state-of-the-art analysis, it is imperative to adopt a training SLM model using the LoRA approach to address the memory overhead during training of large models. Therefore, efficient SLM training aims to balance accuracy and computational cost. In the FL environment, we provide a decentralized scenario that enables a more robust study of non-IID scenarios. Consequently, by integrating the benefits of the data settings, the model converges with a small amount of data and also provides better network flow responses.

Table 1. Related Works.

Work	Benign-only Training	Federated Setting	Network-flow Data	PEFT
Ferrag et al. [Ferrag et al. 2024]	No	No	Yes	No
Maasaoui et al. [Maasaoui et al. 2024]	No	No	Yes	No
Sánchez et al. [Sánchez et al. 2024]	No	No	No	No
Guo et al. [Guo et al. 2021]	Yes	No	No	No
Talasso et al. [Talasso et al. 2025]	Yes	Yes	No	Yes
FL-TFlow (Ours)	**Yes**	**Yes**	**Yes**	**Yes**

3. A Benign-only Fine-tune of SLM for Ransomware Detection on Resource Constrained Scenarios

This section presents FL-TFlow, an end-to-end pipeline for benign-only ransomware detection from IoT/Edge network flows. Specifically, FL-TFlow considers a leakage-controlled flow-to-text representation, FL fine-tuning of a pre-trained SLM using LoRA adapters (with FedProx to mitigate Non-IID client drift), and scoring and decision rules based on token-prediction scores derived from the SLM outputs, including temporal metrics such as time-to-detection and coverage. In this section, we introduce the scenario overview and FL-TFlow Operations.

3.1. Scenario overview

Figure 1 shows the whole workflow from start to finish and points out where privacy, efficiency, and operational evaluation come into play. Each client trains only on benign network flows, allowing the SLM to learn the expected token distributions of normal traffic. When making predictions, anomalies are found when the top-k prediction score drops below a threshold, indicating that the flow is not following learned benign patterns. Clients

only send LoRA adapter updates, not raw data or full model weights. The timestamps and source IP addresses are only used for operational metrics, not as model input.

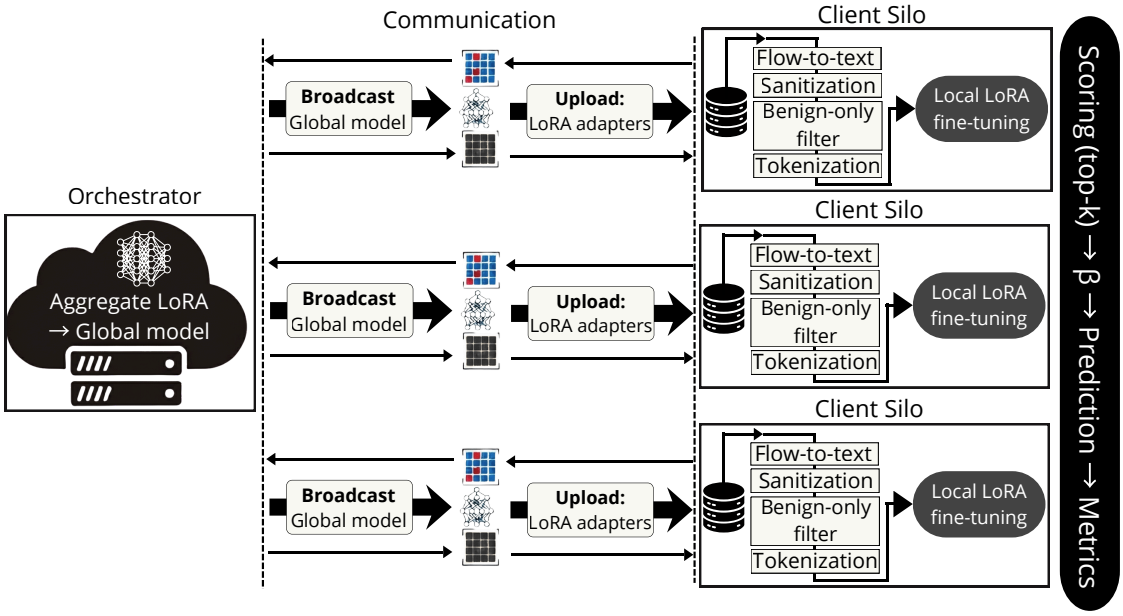


Figure 1. Scenario overview of FL-TFlow

We consider a scenario involving n devices, denoted as $\mathcal{U} = \{u_1, \ldots, u_n\}$, where each device is uniquely identified by an index i within the range of $[1, n]$. In this typical FL setup, each communication round begins by selecting a subset of devices, $\mathcal{C} \subseteq \mathcal{U}$, to receive the global model M_g and to train on their respective local datasets D_i. Each device u_i has a local dataset D_i, which varies significantly in size, features, and label distribution, typically exhibiting non-IID characteristics. The dataset D_i consists of features $x_{k,i}$ for $D \in \{D_1, \ldots, |D_i|\}$, paired with corresponding labels $y_{k,i}$, and $|D_i|$ represents the number of data samples collected by u_i. Each client aims to improve the local model by updating the model parameters to minimize a locally defined loss function using a specified number of epochs E. The server then aggregates the updates via FedAvg, as defined in Eq. 1. Let us denote n_i as the number of benign training samples at the client i.

$$\phi^{t+1} = \sum_{i \in C_t} \frac{n_i}{\sum_{j \in C_t} n_j} \phi_i^{t+1}, \tag{1}$$

An IoT/Edge deployment in which multiple clients (*e.g.*, gateways or edge nodes) collect network-flow telemetry from their local environments. In this sense, each client keeps raw flows locally and participates in a client-server FL process coordinated by an orchestrator. The server keeps a global pre-trained SLM that has been changed with LoRA. This makes sure that only lightweight adapter parameters can be trained and shared. The server randomly selects a group of clients and sends them the current global model in each communication round. Next, selected clients run FL fine-tuning and send back only LoRA adapter updates. The server then combines these updates to create the next global model. The server aggregates the returned LoRA updates to produce the next global model, and it does not have access to the raw client traffic. The network-flow records collected on

IoT/Edge devices or gateways FL-TFlow output an anomaly score and a binary decision per flow (benign vs. ransomware). After training, we compute a score for each test flow using a top-k token-prediction score (legacy_flat) derived from the SLM logits and apply a threshold rule to produce the final decision. Finally, we compute deployment-oriented metrics (TTD/coverage/benign FPR) from predictions using Src IP and Timestamp (metadata not used as model input).

3.2. FL-TFlow Operations

The biggest problem with using language models on network traffic is figuring out how to turn raw flow data into text input. Network flows include numbers (ports, byte counts, durations) that lack meaning in natural language. System logs, on the other hand, are always written in text. Our flow-to-text serialization addresses this by converting each flow into a structured sentence composed of "feature value" pairs. That lets the SLM see how the network works as a series of tokens.

Figure 2 shows how a raw flow changes into a *Content* sequence and where the system makes sure that there is no leakage. Fields that carry labels, like *Attack Name*, are never included in *Content*. That stops simple shortcuts that would make the results invalid. Device/time attributes (e.g., Src IP and Timestamp) are preserved solely for temporal assessment and sanitization, focusing on high-cardinality identifiers while avoiding aggregating numerical magnitudes that represent traffic behavior. We use the pre-trained model tokenizer to tokenize *Content*, add an end-of-sequence token, and ensure the resulting sequence is no longer than a given length. We use a standard autoregressive objective to train the SLM by setting the labels to the input tokens' IDs.

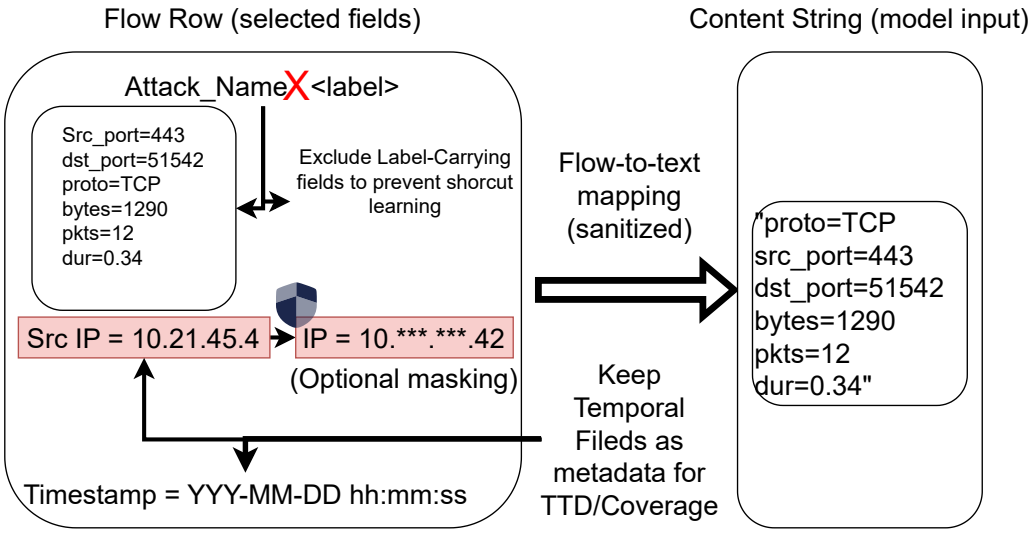


Figure 2. Flow-to-Text serialization with leakage control.

Specifically, the system converts each flow into a compact text sequence of "feature value" tokens. That preserves statistically meaningful traffic behavior while remaining compatible with standard LLM tokenizers. In the current implementation, the *Content* includes protocol, flow duration, timing/periodic behavior (*e.g.*, mean inter-arrival time),

volume/rate descriptors (forward/backward packet counts and byte rate), packet-length statistics (mean/variance), and TCP behavior (*e.g.*, SYN/FIN flag counts).

We consider a client-server FL setup. The server initializes a pre-trained language model and applies LoRA, leaving only a small set of adapter parameters trainable. Each client receives the current global model, performs local fine-tuning on its benign-only data samples for a limited number of steps, and then returns only the LoRA adapter tensors, parameters whose names include `lora_*`. This design reduces both local training overhead and communication payloads.

Formally, for each linear layer adapted with a frozen weight matrix W, LoRA learns a low-rank update ΔW. During training and inference, the model uses $W' = W + \Delta W$, as shown in Eq. 2. Here, r is the adapter rank, and α is a scaling factor. Only the matrices (A, B) are trainable, and the clients and the server exchange only these parameters, following the PEFT approach of LoRA [Hu et al. 2021].

$$W' = W + \frac{\alpha}{r} BA, \quad A \in \mathbb{R}^{r \times d_{\text{in}}},\ B \in \mathbb{R}^{d_{\text{out}} \times r}. \tag{2}$$

During local optimization, clients employ FedProx regularization [Li et al. 2020], adding a proximal term that penalizes deviation from the current global model to improve robustness under heterogeneous, non-IID client telemetry. Let us denote ϕ^t as the global LoRA adapter parameters in round t, and let C_t be the set of selected clients. Each client $i \in C_t$ updates its local adapters by minimizing a benign-only objective combined with FedProx, as shown in Eq. 3. We denote F_i as the client's empirical loss on benign traffic flows, and μ controls the strength of the proximal regularization.

$$\phi_i^{t+1} = \underset{\phi}{\operatorname{argmin}}\ F_i(\phi) + \frac{\mu}{2} \left\| \phi - \phi^t \right\|_2^2, \tag{3}$$

For each test flow, we compute a predictive-quality score based on a top-k token-prediction criterion applied to the tokenized *Content* sequence. Smaller scores indicate a stronger deviation from the patterns learned from benign-only training. We then apply a threshold decision rule: predict ransomware (anomaly) if $score < \beta$; otherwise, predict benign. This matches the behavior of the implemented classifier (`pred = (score < threshold)`). In our experiments, we use the `legacy_flat` scoring variant to match the current evaluation pipeline.

Given a tokenized sequence $x_{1:L}$, let us denote $\text{Top-}k(\mathbf{z}_t)$ as the set of the highest-probability token IDs k in the model logits $\mathbf{z}_t$ at position t. The `legacy_flat` score is defined as the fraction of input tokens that appear anywhere in the union of top-k predictions across the whole sequence, as shown in Eq. 4. L denotes the effective sequence length after truncation/padding (excluding padding positions), and we flag a flow as ransomware when $\textit{score}(x) < \beta$.

$$\textit{score}(x) = \frac{1}{L} \sum_{t=1}^{L} \mathbb{I}\left[x_t \in \bigcup_{j=1}^{L} \text{Top-}k(\mathbf{z}_j) \right], \tag{4}$$

Algorithm 1 details the FL training procedure. Each round consists of three

phases: client selection, local training, and server aggregation. During initialization of our FL system, we might need to select a subset of clients $\mathcal{U}$, meaning a random subset of clients will train in each round. In the next step, each client will, in parallel, send their ϕ^t to the server. However, while each client has its local sample, it uses its training batch to compute its LoRA updates $\mathcal{L}$, which mainly balances the parameters to improve and validate the relevance of each parameter used in LoRA. During local training, each client optimizes the FedProx objective (Eq. 3) using only benign traffic samples, exclusively updating the LoRA adapter weights while keeping the base SLM frozen. This design reduces communication overhead by having clients transmit only the lightweight adapter parameters rather than the full model weights. The server aggregates these updates via weighted averaging to produce the global model for the next round. That sequence repeats in each round of training: the server randomly selects clients, trains them, and computes their losses to validate the LoRA parameters, validate the model, and make improvements.

Algorithm 1: FL-TFlow: Federated Benign-Only Fine-Tuning for Ransomware Detection

Data: Global LoRA adapters ϕ^0, SLM M, total rounds T, client fraction C, client set $\mathcal{U}$, local epochs E, regularization μ

for *round* $t = 0$ ***to*** $T - 1$ **do**
  Client Selection:
  $\mathcal{C}_t \leftarrow$ random subset of $\mathcal{U}$ with size $C \cdot |\mathcal{U}|$
  Send ϕ^t to all clients $i \in \mathcal{C}_t$
  Parallel Client Updates:
  for *each client* $i \in \mathcal{C}_t$ *in parallel* **do**
    $\phi_i^{t+1} \leftarrow \text{LocalTrain}(i, \phi^t)$
    for *epoch* $e = 1$ ***to*** E **do**
      Sample batch b from client i's benign data
      Compute loss: $\mathcal{L} = F_i(\phi_i) + \frac{\mu}{2}\|\phi_i - \phi^t\|_2^2$
      Update only LoRA parameters (names containing `lora_*`)
    end
  end
  Server Aggregation:
  $\phi^{t+1} \leftarrow \text{Aggregate}(\{\phi_i^{t+1} \mid i \in \mathcal{C}_t\})$
end

4. Evaluation

This section presents the experimental setup used to evaluate the performance and efficiency of FL-TFlow under realistic IoT/Edge federated conditions. We describe the simulated FL scenario, dataset, and key parameters, and compare a baseline run against a legacy client-training variant under the same FL and Non-IID setting.

4.1. Experimental Setup

Our code can be found on Github[1]. All experiments were conducted on a server equipped with an Intel Core i9-13900K CPU, 128 GB of RAM, and an NVIDIA GeForce RTX 4090

[1]FL-TFlow is available at `https://github.com/81wallace18/FL_RANSOM_LLM.git`

GPU (24 GB VRAM). Table 2 summarizes the main federated configuration and model hyperparameters used throughout the evaluation.

Table 2. Summary of the experimental setup.

Component	Configuration
Dataset	CIC-BCCC-NRC-Edge-IIoTSet-2022
Data Split	Bening only approach
Model	SmolLM-135M with LoRA adaptation
LoRA Parameters	Rank $r = 16$, $\alpha = 64$ ($= 4r$), dropout 0.1
Tokenization	Max length 1024
Federated Setup	50 clients
Rounds	30
Percentage of clients per round	20%
Local Training	50 steps, batch size 4
Non-IID Setting	hetero_device, Dirichlet ($\alpha = 0.5$)
FedProx	$\mu = 0.0005$
Scoring Metric	Top-k token-prediction score ($k \in \{1, 3, 5, 10\}$)
Dirichlet α	0.5

The simulation involves 50 clients, where we randomly select 20% of them for training in each FL round. Using the SmolLM-135M, we adapt the model with the LoRA method (r=16) and a dropout rate of 0.1, resulting in a model-specific SLM. Our model uses a batch size of 4 samples and a 50-step training epoch to improve understanding. In this way, our execution uses 30 rounds to complete the simulation. The model's maximum tokenization is 1024, so we also use logs with that max to understand the model.

For our experiments, we use the CIC-BCCC-NRC-Edge-IIoTSet-2022 dataset, a well-known collection of IoT network traffic labeled as benign or malicious. We split the dataset into 80% for training and 20% for testing. To simulate a realistic heterogeneous IoT/Edge environment, we assign the training data to 50 FL clients divided into three capacity classes: light (20%), intermediate (40%), and high (40%). Table 3 reports the characteristics of the dataset split, which makes the training only with the benign labels. We sample from that dataset using a Dirichlet distribution with $\alpha = 0.5$, creating a non-IID setting with quantity skew. Higher-capacity clients receive proportionally more training samples. The server selects clients in each training round proportionally to their local data volume and aggregates model updates weighted by the number of samples from each client. We train exclusively on benign traffic from this split, so the training set contains no ransomware samples. In contrast, the test set includes both benign and ransomware samples, enabling evaluation of the model's ability to detect ransomware on unseen data from unique IP addresses.

We compare FL-TFlow against a baseline framework introduced by [Talasso et al. 2025], which performs benign-only federated adaptation of a pre-trained language model using LoRA and exchanges only adapter updates between clients and the server. Moreover, we compare LogBert [Guo et al. 2021] (bidirectional), a

Table 3. Dataset Split Characteristics

Attribute	Train	Test
Total Samples	65,322	16,970
Benign (Label=0)	65,322	16,331
Ransom (Label=1)	0	639
Unique Src IP (Total)	13	14
Unique Src IP (Benign)	13	13
Unique Src IP (Ransom)	0	1

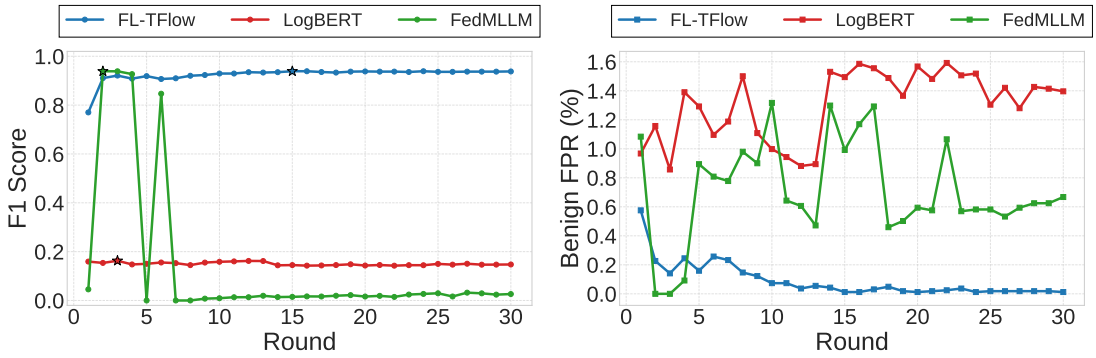
BERT-based model designed to learn what constitutes "normal" behavior by randomly masking tokens in system logs and training the model to predict them. Additionally, it enforces the representations of normal log sequences to lie close to each other within a hypersphere. Originally developed for system logs (such as HDFS), we adapted it for our scenario to ensure a fair comparison. We retrained the model on the same network flow dataset using FedAvg with full-parameter fine-tuning. In FL-TFlow, we transfer this benign-only federated LoRA setup to IoT/edge network flows by using a leakage-controlled flow-to-text representation and FedProx regularization to mitigate client drift under Non-IID data. We report both detection performance and deployment-oriented metrics, including benign false-positive rate and time-to-detection.

We consider two well-known metrics to measure system performance, namely, top-k token prediction F1 and FPR. Specifically, the top-k token prediction accuracy measures, for each tokenized flow, the fraction of input tokens that appear among the model's k most probable predictions across the entire sequence. High scores indicate that the model recognizes the flow as consistent with the learned benign traffic patterns, while low scores indicate anomalous deviation. F1 score balances both precision and recall, ensuring the model reliably detects ransomware while minimizing false alarms. A high F1 score indicates that the system can accurately identify threats without overwhelming security teams with false alarms. Precision measures the reliability of ransomware alerts. In this sense, high precision means it is likely correct, as soon as the system flags a flow as malicious, reducing wasted investigation time. Recall measures the system's ability to capture all ransomware flows. In this sense, high recall ensures that few attacks go undetected, which is critical for preventing data encryption and service disruption in IoT environments.

FPR evaluates the proportion of instances in which an identity claim was false. However, the system mistakenly accepted it, *i.e.*, it incorrectly grants access to unauthorized users, leading to false acceptances. A low FPR is crucial in ransomware detection because it minimizes the risk of blocking legitimate traffic, which could disrupt normal network operations. In IoT/Edge environments, false alarms can lead to unnecessary service interruptions and reduce user trust in the detection system. For TTD, we evaluate at both flow and window levels (30s). Flow-level analysis considers the precision of classifying individual network connections, while the 30s window evaluates the efficacy of identifying broader attack patterns by aggregating flow data into fixed intervals.

4.2. Results

Figure 3 shows the training-round evaluations of F1 Score and FPR on Benign-only data for the 30 rounds of our evaluated tree methods. Figure 3(a) shows the F1 Score evaluation of the scenario we determined for the benign-only method. In this way, we have the poor method, LogBERT, which proposes a method with some failures due to its pretrained methodology compared to ours. At the same time, LogBERT was originally trained on a different capability of reading the full log. When dealing with directly filtered logs, it has some issues determining the syntagms with its previous training, reaching F1 scores around 0.15 during the rounds, because its training capabilities do not consider filtered logs.



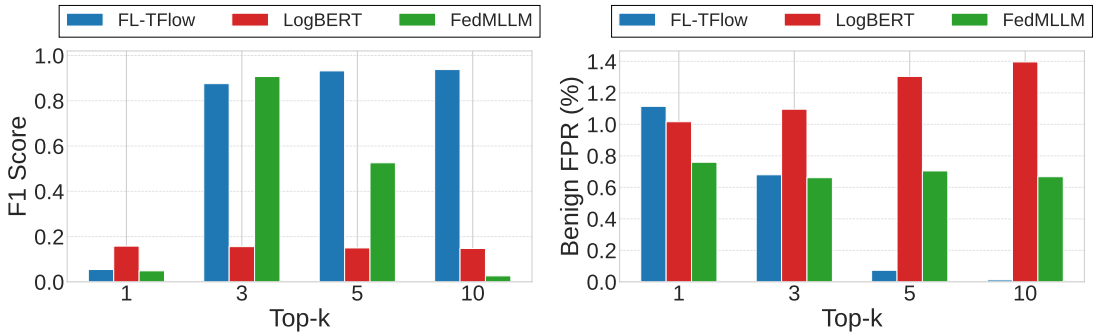
(a) F1 Score (b) Benign FPR

Figure 3. F1 Score and Benign FPR for the evaluated frameworks over 30 rounds. The star indicates the best point on each curve

Now, in FedMLLM (Figure 3(a)), we have a more similar experience to our training. However, due to this approach not considering a full training of the LoRA parameters using a quantized LoRA, the evaluation shows that the inference of the model does not converge very well in different datasets, for which FedMLLM was not trained, because of that facilities the best F1 score was in the second round by reaches above of 0.18 and mantaining it during the rest of the rounds due the quandized values of 4bits in the inference step dificult in adapt in our simulation. Now our method the FL-TFlow shows the best evaluation at the round 15 reaches above 0.93 of F1 score that occours due the adaptation of our LoRA updates using the score showed at Eq. 4. That adaptation helps the model to improve the evaluation using minnor amount of tokens and also helping it to improve its accuracy dealing with the integration of LoRA parameters and FedProx to adapt the parameters during each round.

Figure 3(b) shows the FPR evaluation, which across all simulated methods reaches a common low value, never exceeding 1.7%. In that way, the LogBERT is similar to the F1 score, also because it uses pre-trained parameters with too many tokens, resulting in higher evaluation scores that fluctuate during the rounds, with round 12 reaching 0.9% but then increasing the FPR. Also, a similar pattern is observed during rounds with the FedMLLM, with fluctuating values that peak at round 13 (0.48%), because the training does not converge during the simulation. FL-TFlow achieves the best FPR in this scenario, with a convergence step that decreases throughout all rounds and a small standard deviation of these values, indicating convergence.

Figure 4 shows the evaluation of the top-k parameter at the final round 30 using our tree methods, comparing F1 score and FPR. Figure 4(a) compares the F1 metric scores at 1, 3, 5, and 10 top-k. These scores reflect the parameter adaptation criteria, indicating that the solutions improve the method described by Eq. 4. At top-k 1, FedMLLM and FL-TFlow achieve similar F1 scores of 0.05. Logbert is the best at the moment, with an F1 score of 0.16, because its specialization in large logs helps maintain its accuracy. Now, after passing to top-k 3, LogBERT loses relevance and maintains it in the next top-k values due to its poor evaluation at the moment. However, FedMLLM has the best F1 score, reaching almost 0.9, while FL-TFlow reaches 0.88. However, while we have a bigger k, the F1 score of FedMLLM decreases while the FL-TFlow increases, reaching respectively at k 5 and 10: 0.53 and 0.03, while FL-TFlow reaches 0.93 and 0.94. That occurs because FL-TFlow guarantees the convergence of the model and its parameters, whereas the FedMLLM does not account for this, leading to 4-bit quantization that is too low.



(a) F1 Score (b) Benign FPR

Figure 4. Sensitivity to Top-k parameter at Round 30.

Figure 4(b) shows the FPR evaluation at the top-adjustment using our tree methods. That way, we can validate the value of top-k as the model converges on the classification. At the top-k 1, we can see that our method FL-TFlow shows a higher FPR and a poor evaluation, exceeding 1.1%, while LogBERT is 0.9% and FedMLLM 0.78%. However, after that value, we observe the parameters' convergence decrease by 3, as does FL-TFlow, and FedMLLM shows similar values at 0.7 and 0.68%. However, LogBERT fails to achieve classification convergence, leading to a higher FPR. At top-k 5 and 10, LogBERT increases its error to 1.3 and 1.38 FL-TFlow, respectively, and shows a significant decrease in classification convergence for ransomware, reaching above 0.1% and maintaining it in the top-k 10 with almost 0 of FPR. In contrast, the FedMLLM maintains its FPR behavior above 0.7%.

Figure 5 shows our operational calibration of the method. We put our evaluation method into practice after training, using top-k 10 to compare the Flow at the time in the network with a 30-second window to respond. First, when using our tree methods in the Flow, FL-TFlow achieves an F1 score of 0.94 in this scenario, indicating it is well-suited for real-time environments that do not require a high time-to-respond. While in Flow, FL-TFlow achieves the best F1 score, while LogBERT reaches 0.15 due to the large number of logs it expects. In that way, FedMLLM reaches 0.02 by treating its 4-bit values as trainable parameters.

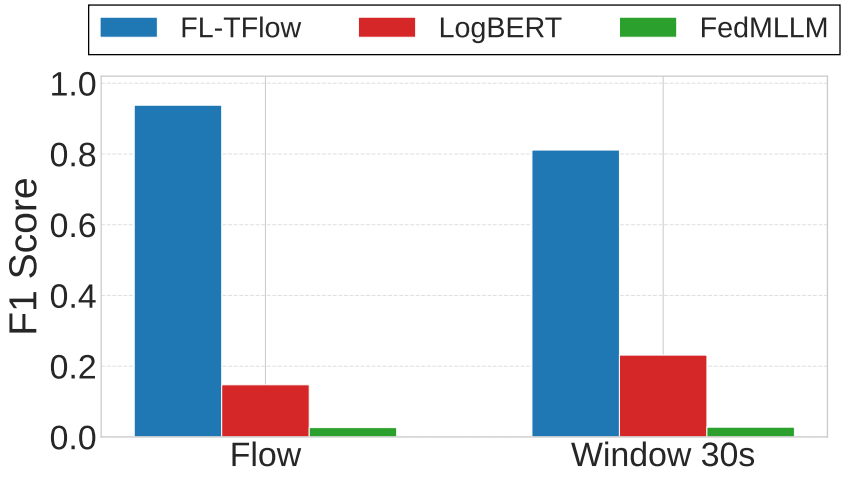


Figure 5. Flow vs. window(30s) F1 at Round 30 ($k = 10$), operational calibration.

While the Figure 5 shows the Flow having the best effort of FL-TFlow in this scenario, in the window of 30s, there is a decrease in F1 score, reaching above 0.8, which occurs because this method does not have an optimization to respond faster, as it uses quantization on the inference layers. In that way, the LogBERT shows an increase due to the 30s windows receiving too much data in time, and improves the F1 to 0.22, which shows the preparation of this treatment on the pre-trained parameters for that approach. However, their training evaluation was insufficient to achieve high scores in this simulation. Because the top-k method does not offer an efficient training for FedMLLM, it achieves the same evaluation in both scenarios at 0.02.

5. Conclusion

This paper presents FL-TFlow, a collaborative AI system designed to spot ransomware in intelligent devices and edge networks. The system addresses a significant challenge: how to accurately monitor threats when data is private and distributed across many locations, making it impossible to aggregate in a single central location. It allows different devices to work together by learning only from regular, safe network traffic-so it never needs to see or share real attack data. At its heart, FL-TFlow uses a safe-traffic-only detection process.

In future work, we plan to improve FL-TFlow to handle evolving attack patterns by automatically adjusting its sensitivity. We also aim to test it against more types of ransomware and in busier, more unpredictable network conditions. Finally, we will explore how different ways of processing network data affect the balance between speed, detection delay, and the computing power needed on small devices.

Acknowledgements

This work was supported by the Ministry of Science, Technology and Innovation, with resources from Law No. 8,248 of October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex and published as Cognitive Architecture (Phase 3), DOU 01245.003479/2024-10. This research is also part of the INCT of Intelligent Communications Networks and the Internet of Things (ICoNIoT) funded by CNPq (proc. 405940/2022-0) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 88887.954253/2024-00. Furthermore, this work was

funded by the National Institute of Science and Technology in Artificial Intelligence Applied to Smart and Sustainable Cities in the Brazilian Amazon (INCT IAmazônia), grant nº 409001/2024-4. Finally, the authors would like to acknowledge the support from CNPq under grant number 405940/2022-0. This paper was supported by a project funded by the Information and Communication Technology Company of the State of Pará (PRODEPA) under process number 2024/0530503.

References

Alsuwaiket, M. A. (2025). Zeroday-llm: A large language model framework for zero-day threat detection in cybersecurity. *Information*, 16(11):939.

Buyuktanir, B., Altinkaya, Ş., Karatas Baydogmus, G., and Yildiz, K. (2025). Federated learning in intrusion detection: advancements, applications, and future directions. *Cluster Computing*, 28(7):473.

Chen, S. and Liao, H. (2022). Bert-log: Anomaly detection for system logs based on pre-trained language model. *Applied Artificial Intelligence*, 36(1).

Ferrag, M. A., Ndhlovu, M., Tihanyi, N., Cordeiro, L. C., Debbah, M., Lestable, T., and Thandi, N. S. (2024). Revolutionizing cyber threat detection with large language models: A privacy-preserving bert-based lightweight model for iot/iiot devices. *IEEE Access*, 12:23733–23750.

Guo, H., Yuan, S., and Wu, X. (2021). Logbert: Log anomaly detection via bert. In *International Joint Conference on Neural Networks (IJCNN)*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. (2021). Lora: Low-rank adaptation of large language models.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated optimization in heterogeneous networks. *Machine learning and systems*, 2:429–450.

Maasaoui, Z., Battou, A., Merzouki, M., and Lbath, A. (2024). Anomaly based intrusion detection using large language models. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*. NIST.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pages 1273–1282. PMLR.

Popoola, S. I., Ande, R., Adebisi, B., Gui, G., Hammoudeh, M., and Jogunola, O. (2021). Federated deep learning for zero-day botnet attack detection in iot-edge devices. *IEEE Internet of Things Journal*, 9(5):3930–3944.

Sánchez, P. M. S., Huertas Celdrán, A., Bovet, G., and Martínez Pérez, G. (2024). Transfer learning in pre-trained large language models for malware detection based on system calls. In *IEEE Military Communications Conference (MILCOM)*, pages 853–858.

Talasso, G. U., de Souza, A. M., Guidoni, D., Cerqueira, E., and Villas, L. A. (2025). Finetuning eficiente de modelos de linguagem para detectar anomalias em logs privados usando aprendizado federado. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Vailshery, L. S. (2024). Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2033. *https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/*. Accessed: 2025-01-30.