FL-SVM: A Federated Learning-Based Support Vector Machine Model for IoT Malware Detection

DOI: https://doi.org/10.54654/isj.v2i28.1243

Nguyen Ngoc Toan*, Nguyen Manh Tuan

***Abstract* - The rapid development of IoT devices has significantly contributed to digital transformation across organizations, enterprises, and institutions. The risk of malware infection on IoT devices has become increasingly prevalent and dangerous, with new attack methods and infection techniques. IoT devices, with their numerous, diverse types, configurations and resource usage characteristics, have raised new requirements for more efficient, accurate IoT malware detection methods and solutions that ensure privacy during model training in real-world applications. In this paper, we propose a more efficient IoT malware detection model based on an improved Federated Learning method. Specifically, our key contributions include a dynamic aggregation mechanism designed for clients with heterogeneous feature spaces, allowing resource-constrained IoT devices to adaptively adjust their feature dimensionality according to hardware capacity. The proposed malware detection model has been tested with an IoT dataset on the MIPS architecture platform. Experimental results show that the proposed malware detection model achieves good accuracy while strongly leveraging the advantages of Federated Learning in ensuring data privacy and minimizing computational resource usage during model training.**

***Tóm tắt* - Sự phát triển của các thiết bị IoT đã và đang góp phần quan trọng hỗ trợ chuyển đổi số tại các cơ quan, tổ chức, doanh nghiệp. Các nguy cơ lây nhiễm mã độc trên thiết bị IoT ngày càng trở lên thường trực và nguy hiểm hơn với các hình thức tấn công, kỹ thuật lây nhiễm mới. Các thiết bị IoT với các đặc trưng về số lượng thiết bị nhiều, đang dạng về chủng loại, cấu hình, tài nguyên sử dụng đã đặt ra nhiều yêu cầu mới về phương pháp, giải pháp phát hiện mã độc IoT hiệu quả hơn, chính xác hơn, đảm bảo tính riêng tư trong huấn luyện mô hình trong thực tế hiện nay. Trong bài báo này, chúng tôi đề xuất mô hình phát hiện mã độc IoT hiệu quả hơn dựa trên cải tiến phương pháp Federated Learning. Đặc biệt, đóng góp chính của chúng tôi bao gồm cơ chế tổng hợp động cho các máy khách có không gian đặc trưng không đồng nhất, cho phép thiết bị IoT thích ứng số lượng đặc trưng theo cấu hình phần cứng. Mô hình phát hiện mã độc đề xuất đã được thử nghiệm với tập dữ liệu IoT trên nền tảng kiến trúc MIPS. Các thực nghiệm đã chứng minh mô hình phát hiện mã độc đề xuất cho độ chính xác tốt, đồng thời phát huy mạnh mẽ ưu điểm của phương pháp Federated Learning trong đảm bảo tính riêng tư dữ liệu và hạn chế tài nguyên tính toán trong huấn luyện mô hình.**

Keywords—IoT Malware; federated learning; machine learning; artificial intelligence; malware detection.

Từ khóa— Mã độc IoT; học liên kết; học máy; trí tuệ nhân tạo; phát hiện mã độc.

I. INTRODUCTION

The development of the Internet of Things (IoT) has been reshaping the global technology landscape. According to IHS Markit [1], it is estimated that by 2025, more than 75 billion devices will be connected to the internet across various sectors of life and society. However, along with rapid development and the increasing need for intelligent processing, IoT devices are designed with limited hardware configurations to optimize manufacturing costs and energy consumption. In fact, IoT devices commonly found in Vietnam, such as the TP-Link WR841N router, are equipped with a 400 MHz CPU and 32MB of RAM, which is much less powerful compared to modern smartphones or computers

This manuscript was received December 10, 2025. It was reviewed on May 4, 2026, revised on July 27, 2026 and accepted on August 20, 2026.

* Corresponding author

[2]. According to a survey by Shodan.io, tens of thousands of devices with similar configurations are operating on the internet without adequate protection measures. Sivanathan et al. [3] also pointed out that more than 70% of IoT devices in household environments have less than 64MB of RAM and are not updated regularly for security. These factors have made these devices common targets for cybercriminals, as evidenced by the Mirai botnet, which infected over 600,000 IoT devices and carried out the largest DDoS attack in 2017 [4]. In 2019, its variants conducted attacks that took down over 900,000 Deutsche Telekom routers [5].

To detect malware on resource-constrained IoT devices with diverse architectures, some researches have focused on using traditional lightweight machine learning models such as Decision Trees [6], Random Forest [6] or Support Vector Machines [7-9] with the goal of sacrificing some accuracy to enable deployment on resource-constrained devices. Tien et al. [6] proposed methods achieving up to accuracy 98% with Random Forests, Naive Bayes, and K-Nearest Neighbors, using opcode and ELF features. Toan et al. [7] proposed an effective model by applying the TF-IDF method combined with traditional machine learning algorithms using opcode sequence features. Ramamoorthy et al. [9] proposed a model with accuracy 92% by combining machine learning algorithms with system call features. Dung et al. [8] used traditional machine learning and multi-architecture opcode conversion methods to effectively detect IoT malware. Other studies have taken an optimization approach by applying techniques such as pruning to remove unimportant connections, quantization to reduce numerical representation precision, or knowledge distillation to transfer knowledge from a large model to a smaller one. While these improved methods have initially shown effectiveness in malware detection, most current solutions are still based on centralized models, which require data collection from devices to a server for training and model updates. Privacy concerns, bandwidth issues in network systems during real-world deployment, and the handling of these challenges have not been sufficiently addressed.

Malware detection on IoT devices presents challenges that traditional machine learning models have not effectively solved due to heterogeneity in hardware architecture, operating systems, and communication protocols across device manufacturers. The application of modern deep learning models such as Convolutional Neural Networks or Long Short-Term Memory, while improving accuracy, faces issues with resource constraints and processing speed in IoT devices. Ramamoorthy et al. [10] proposed a malware detection model based on LSTM achieving accuracy 89.2%, and 91.93% with a textCNN-based model. In another study, Ramamoorthy et al. [9] proposed a model that achieved accuracy 92.07% with an MLP architecture. Tien et al. [6] proposed a Deep learning model using opcode features, capable of detecting malware with accuracy 98%. However, compared to traditional machine learning models such as SVM, deep learning approaches generally require significantly higher computational resources, including memory consumption and training time. This limitation makes them less suitable for deployment on resource-constrained IoT devices. In contrast, traditional machine learning models provide a more favorable trade-off between detection performance and computational efficiency, making them more practical for real-world IoT environments.

Federated Learning has been researched as a solution to address malware detection in distributed and resource-constrained IoT environments. Unlike traditional centralized machine learning models, Federated Learning allows devices to participate in local model training on their private data and only share updated model parameters with the coordinating server [11]. The server then aggregates and updates the model to create a more effective global model, which is redistributed to the devices for the next training round. This approach not only preserves data privacy on each device during training but also reduces communication costs compared to sending all raw data to a central server. Moreover, distributed training can leverage the combined computational power of individual IoT devices. However, current research on Federated

Learning for IoT mainly focuses on optimizing aggregation algorithms or reducing transmission costs, with little research dedicated to solving the issue of feature representation suitable for resource-constrained IoT devices. Studies on Federated Learning for IoT malware detection primarily focus on ARM platforms using Android OS, which have much richer resources compared to other IoT devices in the network system, such as MIPS-based devices.

Based on this reality, we propose a new IoT malware detection model based on the Federated Learning combined with Support Vector Machines (FL-SVM). The following are the two main contributions of this research:

- Proposing a dynamic federated learning aggregation algorithm for heterogeneous feature spaces, allowing adaptive feature selection based on individual IoT device hardware constraints while ensuring high accuracy, data privacy, and minimal bandwidth usage.

- Evaluating the Federated Learning method and feature extraction techniques on the dataset (the MIPS dataset consisting of 4,393 malware files and 4,393 clean files).

II. Related Work

In recent years, malware detection methods using static features have been widely applied to detect malware on resource-constrained IoT devices running Linux-based operating systems, with high effectiveness and low complexity. Commonly used static features include file headers, strings, opcodes, system calls, or other static information that does not require the execution of the file. Toan et al. [7] proposed a malware detection model using opcode sequence features, achieving a detection rate of 99.8% and a classification accuracy of 95.8%, with low computational cost, making it suitable for deployment on IoT devices. However, the study only experimentally identified 20 effective opcode features for building a malware detection model for the MIPS architecture. Tien et al. [6] proposed a machine learning model that combines multiple static features, such as ELF headers, opcode histograms, and syscall information extracted from ELF, which performed well on over 6,000 IoT malware samples, supporting multiple IoT architectures (MIPS, ARM, x86). However, this method suffers from reduced accuracy and inefficiency when ELF files are obfuscated or stripped of symbols. Ramamoorthy et al. [9] proposed an approach that extracts static syscalls from ELF binaries on ARM by classifying syscalls by risk level and using machine learning models (Logistic Regression, Random Forest, MLP, etc.). Experimental results demonstrated the effectiveness of the approach with an F1-score of 96.86%. However, this method is not effective when ELF files are stripped or packed. Dung et al. [8] proposed the CAIMP method, which allows learning and converting opcodes across different CPU architectures to build a multi-architecture malware detection model on resource-constrained IoT devices. The detection model based on this approach can achieve an accuracy of 99%. However, the method faces challenges when training the feature conversion model and computing on large datasets.

Federated Learning has proven effective in training malware detection models in a distributed environment without the need for transmitting raw data. Rey et al. [12] demonstrated the effectiveness of Federated Learning in IoT malware detection by training models such as MLP and autoencoders from network data on multiple devices. Çıplak et al. [13] used a DNN architecture in Federated Learning to detect and classify malware with good accuracy and data privacy protection. However, these studies mainly use dynamic features such as network traffic or logs, and collecting feature information faces challenges in multi-architecture IoT environments with resource constraints.

Additionally, the Federated Learning architectures proposed by the authors are designed with deep learning models such as DNN or MLP, which are not well-suited for resource-constrained IoT devices. Nobakht et al. [14] pointed out that integrating Federated Learning into IoT systems requires careful attention to model lightweightness, communication latency, and energy

consumption. A promising direction recently discussed is Tiny Federated Learning, combining Federated Learning with ultra-lightweight models, as proposed by Asiri et al. [15]. The authors suggested redesigning models and update protocols to better fit IoT infrastructure. Furthermore, the FedHGCDroid study [16] introduced a model combining CNN and GNN in Federated Learning for Android malware classification and emphasized adaptation to heterogeneous data and security. Chen et al. [17] proposed an approach integrating heterogeneous data in Federated Learning for malware classification. The study focused on improving the distributed model to handle data and model diversity among clients, thus enhancing accuracy without the need to share raw data. Similarly, Lin and Huang [19] applied basic Federated Learning to malware classification, focusing on the benefits of protecting distributed data and reducing the risk of data leaks, suitable for high-security systems. Hsu et al. [18] introduced a Federated Learning system that protects privacy for Android malware detection based on edge computing, where edge devices can train local models without uploading data to the cloud, reducing latency and enhancing security. Additionally, Taheri et al. [20] developed the Fed-IIoT architecture – a robust Federated Learning framework dedicated to IoT malware detection in industrial IoT, focusing on handling Android data and defending against attacks while maintaining high performance in distributed environments. These studies have introduced new directions for malware detection on resource-constrained IoT devices but still mainly rely on data collected from dynamic analysis or complex feature combinations. Moreover, the models have not fully addressed the issue of lightweight models suitable for widespread deployment on devices with limited memory and resources.

Therefore, static feature-based malware detection methods provide high effectiveness, accuracy, and practicality for deployment on IoT devices due to their low computational cost. However, most methods still follow a centralized approach that does not ensure privacy. In contrast, Federated Learning-based studies provide a robust distributed framework with data privacy protection, but they have not fully explored the effective use of static features and often use complex deep learning models that are difficult to apply to real-world IoT systems. This study, therefore, proposes a combination of static features with a lightweight Federated Learning model to develop a more effective IoT malware detection model in a distributed and resource-constrained environment.

III. Proposed Method

A. Overview of the proposed method

In a distributed IoT environment consisting of n client devices $\{C_1, C_2, \dots, C_n\}$, each device C_i owns, a local dataset D_i, a distinct feature space F_i. Due to differences in computational capacity and memory, devices select a varying number of features from the same global opcode dictionary. Therefore, $|F_i| \neq |F_j|$ and $F_i \cap F_j \subset F_i \cup F_j$ for $i \neq j$. The goal is to find the optimal solution for the global model w^* that minimizes the overall loss function as presented in Eq. (1):

$$w^* = arg \min_w \sum_{i=1}^{n} \frac{|D_i|}{\sum_j |D_j|} . \mathcal{L}_i(w) \qquad (1)$$

where $\mathcal{L}_i$ is the local loss function at device C_i.

Traditional aggregation methods like FedAvg require all clients to have the same model architecture, thus making it impossible to handle cases where the feature spaces differ between devices. Furthermore, data imbalance across clients can cause certain local models to dominate the aggregation process, reducing the fairness and efficiency of the global model. To address these limitations, we propose the development of an IoT malware detection model based on static features of executable files and federated learning. This model aims to solve core issues of IoT environments, including limited resources, data heterogeneity, and privacy protection requirements.

IoT devices exhibit diverse microprocessor architectures, such as Intel, ARM, and MIPS, and machine code structures in ELF files are also highly varied. To standardize the data, each file is

reverse-engineered into assembly code, and static opcode sequences are extracted. These opcode sequences are then represented as numerical vectors using the TF-IDF method to facilitate training the IoT malware detection model.

In the federated learning environment, local data at each device varies significantly, as each device only observes a small subset of malware types. Additionally, each device has its own constraints regarding computational and memory resources. As a result, clients select different feature subsets, both in terms of quantity and content. This heterogeneity makes traditional aggregation methods unsuitable, as they assume all clients have identical feature spaces. Therefore, we propose a dynamic aggregation algorithm capable of handling local models with non-identical feature spaces. Instead of aggregating weights based on fixed positions, the proposed algorithm determines the aggregation weights for each feature based on the size of the local dataset and the frequency of that feature's occurrence across clients. This approach helps ensure that more common features are learned more deeply, while rare features are retained with appropriate weight adjustments, thus ensuring that the global model comprehensively reflects the feature distribution across the system.

The proposed IoT malware detection model is illustrated in Figure 1.

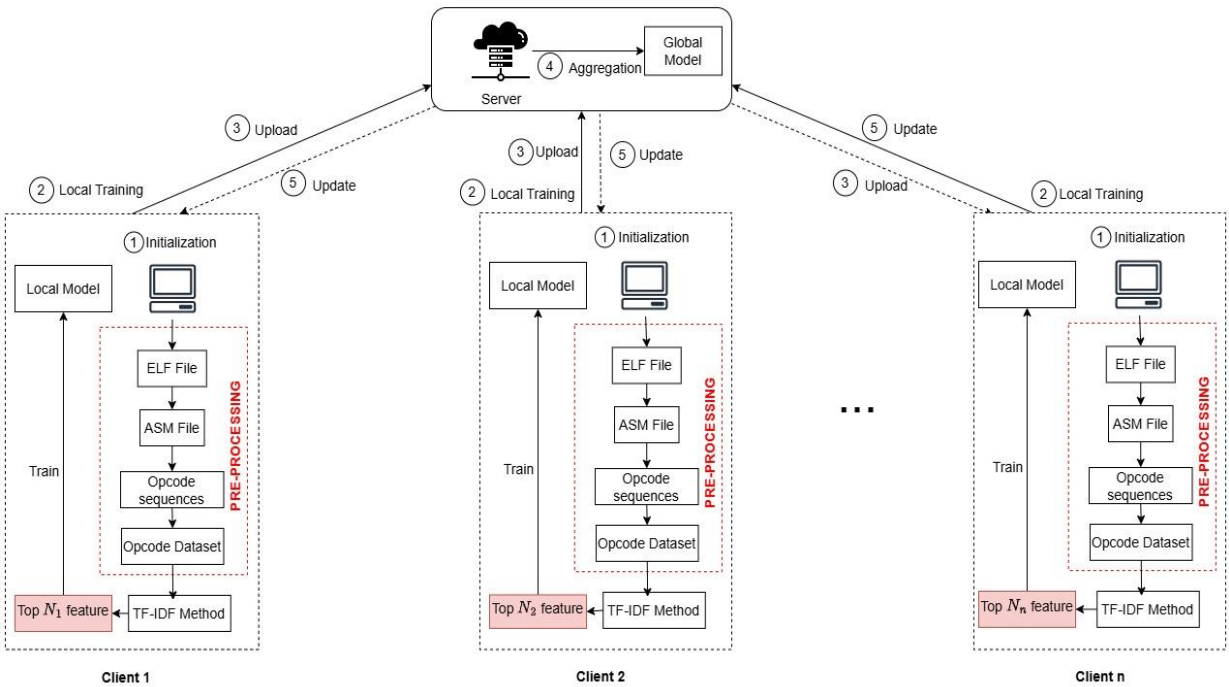

Figure 1. Overall diagram of the proposed IoT malware detection model

The training process is organized into iterative cycles with four main stages, each designed to optimize learning efficiency and minimize resource costs:

Stage 1 - Initializing the global model: The central server initializes the machine learning model with random parameters or predefined values to ensure consistency throughout the distributed training process.

Stage 2 - Local training with different feature counts: Each client trains the model using its own local data. Devices with limited resources select a small subset of important features, while more powerful devices use a larger feature set, optimizing learning efficiency based on available resources.

Stage 3 - Sending Model Parameters: After training, clients send their model weights and biases for the selected features to the server, rather than transmitting the raw data, reducing bandwidth usage and ensuring privacy protection.

Stage 4 - Global model aggregation: The central server uses the proposed aggregation

algorithm to merge the parameters from the clients, map them to the global feature space, and update the shared model. This updated model is then redistributed to the clients for the start of a new iteration.

The federated learning module plays a central role in aggregating information from heterogeneous IoT devices. Unlike traditional methods that assume homogeneous models, this module is designed to adapt flexibly to differences in feature spaces and hardware resources, thereby improving global learning efficiency, maintaining fairness across clients, and ensuring scalability and data security in real-world IoT environments.

The proposed framework follows a horizontal federated learning setting with heterogeneous feature spaces across clients. Each device independently constructs its local TF-IDF feature set from a shared opcode dictionary, leading to partially overlapping but non-identical representations. To address this, a feature-aware dynamic aggregation mechanism is introduced, aligning and combining parameters based on feature identity instead of fixed positions. In addition, the method supports resource-adaptive learning, allowing constrained devices to use fewer features while more capable devices exploit richer representations.

In terms of privacy, the approach ensures that raw data never leaves local devices during training. Sensitive information such as opcode sequences and ELF file contents is processed and stored locally, reducing the risk of data leakage. Only model parameters (weights, biases, and feature indices) are transmitted to the central server for aggregation. This distributed training paradigm minimizes centralized data exposure while still enabling collaborative model improvement across clients.

B. Feature extraction and preprocessing for federated learning

1. Opcode Extraction

Opcode extraction is performed on ELF executable files collected from IoT devices. In this study, the reverse engineering tool IDA Pro is used to disassemble the ELF files and generate the corresponding assembly (.asm) files. From these files, the system extracts opcode sequences that represent the behavior of the executable file. The resulting opcode sequence will serve as input for subsequent processing steps. The opcode extraction procedure is outlined in Algorithm 1.

Algorithm 1: Extracting Opcode Sequence

Input:

- Disassembled file F

- Opcode dictionary D corresponding to architecture

Output: Opcode sequence S

1: Initialize $S \leftarrow \emptyset$

2: Open file F

3: For each line $\ell \in F$ do

4: Extract opcode τ from line ℓ

5: If $\tau \in D$ then

6: Append τ to S

7: End if

8: End for

9: Return S

2. Feature vectorization with TF-IDF and feature selection

Feature vectorization is the step where the text-based opcode sequence is transformed into a statistically meaningful numerical feature vector, while also performing adaptive feature selection to optimize resource usage on each IoT device. We select the Term Frequency-Inverse Document Frequency (TF-IDF) method to represent the importance of each opcode within the entire dataset. This method has been shown to be effective in IoT malware detection [7].

In this study, TF-IDF plays a crucial role in improving the discriminative capability of opcode-based feature representations. Specifically, it emphasizes infrequent but informative opcodes that are more likely to characterize malicious behaviors, while reducing the influence of commonly occurring but less meaningful opcodes. Compared to traditional vectorization approaches such as Bag-of-Words

or raw frequency-based representations, TF-IDF incorporates global statistical information, thereby providing a more balanced and representative feature weighting scheme. Furthermore, the resulting sparse vectors are particularly suitable for federated learning settings, as they contribute to reducing storage requirements and communication overhead during parameter exchange between clients and the central server.

3. Dimensionality reduction

The application of TF-IDF typically generates large and sparse feature matrices, which can consume substantial memory resources and processing time-unsuitable for IoT devices with limited resources. To address this issue, we apply feature selection techniques to reduce the dimensionality of the data. In our experiments, adaptive feature selection can be configured manually based on device resource tiers, the max_features parameter in the TF-IDF vectorizer is used to limit the maximum number of features, retaining only the opcodes with the highest TF-IDF values.

In the federated learning environment, feature selection is performed adaptively: devices with more powerful configurations can select a larger number of features (e.g., the 50 most important features), while resource-constrained devices select a smaller set (e.g., 30 features). However, this leads to different feature spaces across clients, making model aggregation in traditional federated learning challenging. Even when each client independently applies TF-IDF on its local data but uses the same max_features value to synchronize the feature vector size, the actual feature sets of the clients still overlap only partially due to differences in data distribution. The federated learning algorithm proposed in this study addresses this discrepancy through a dynamic aggregation mechanism, ensuring that the global model maintains high performance despite heterogeneity in feature sets and device resources.

C. Training process for the IoT malware detection model based on Federated Learning

1. Model initialization

The first stage involves initializing the global machine learning model, which serves as the foundation for subsequent training rounds. Given the constraints on CPU, memory, and storage resources of IoT devices, the chosen model architecture must be compact yet still ensure high classification performance.

In this study, we propose initializing the model uniformly across all clients with empty initial parameters. This uniform initialization ensures that all clients start from a common global model state, facilitating consistent model aggregation. Simultaneously, training hyperparameters such as learning rate, number of epochs, and optimization algorithm are set during this stage. These parameters are selected to balance model performance and the practical deployment on resource-constrained IoT devices. Once initialization is complete, the model is distributed to the clients for local training.

Local training at clients

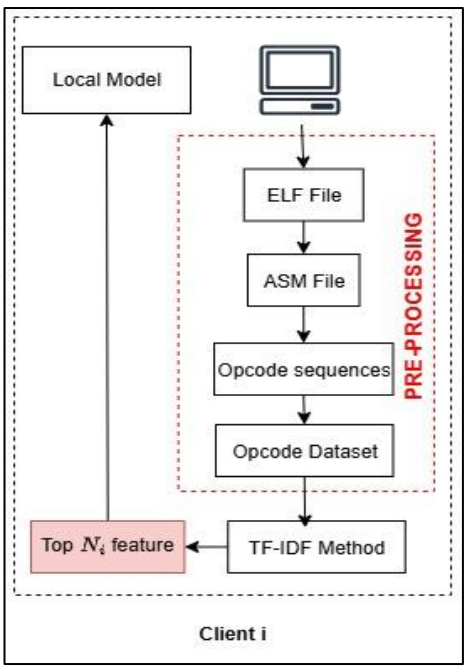

Figure 2. Local Model

The local model structure and training workflow at each client device are illustrated in Figure 2. At each client device, training begins by extracting opcode sequences from ELF executable files to create a local dataset. This data reflects the unique characteristics of each

device due to differences in hardware architecture and malware.

The opcode sequences are converted to numerical representations using the TF-IDF method, with the dimensionality limited by the $\max\_features = K$ parameter to reduce computational complexity. This method automatically removes infrequent opcodes and retains only the more common features that significantly contribute to the classification process.

On the vectorized dataset, a lightweight classifier model is trained to address the binary classification task. Once training is completed, the local parameters, including weight coefficients, bias terms, and the feature set used, are sent to the central server for global model aggregation.

2. Sending parameters to the global model

After each local training round (round t), client k does not send raw data or opcodes but instead sends a parameter package containing: (i) the feature index set I_k used by client, (ii) the corresponding weight coefficients $\left\{\theta_k^{(j)}\right\}_{j\in k}$, (iii) the bias coefficient b_k and (iv) the number of training samples S_k.

To reduce communication overhead, the packet only includes pairs of (index, value) for the present features, making the communication complexity proportional to $|I_k|$ rather than D as defined in Eq. (2):

Packet Structure Sent

$$round\_id = t$$

$$client\_id = k$$

$I_k = [j_1, j_2, \dots]$: List of feature indices

$$theta\_vals = [\theta_k^{j_1}, \theta_k^{j_2}, \dots] \quad (2)$$

$$b_k : intercept$$

S_k : number of training samples

The process of sending parameters is outlined in Algorithm 2.

Algorithm 2: ClientUpload(k, t).

Input:

- Local model at client k: (θ_k, b_k), $\theta_k = \left\{\left(j, \theta_k^{(j)}\right) | \, j \in I_k\right\}$.

- Feature index set I_k.

- Local sample size S_k.

- Round index t.

Output: Payload P sent to server

1: Sort I_k in ascending order

2: Construct $\theta_{val} \leftarrow \left[\theta_k^{(j)} | \, j \in I_k\right]$

3: Construct payload: $P \leftarrow \{ t, k, I_k, \theta_{vals}, b_k, S_k\}$

4: Send P to server

5: Wait for ACK

6: If timeout occurs then

7: Retry sending up to R times

8: End if

9: Return

3. Model aggregation

Upon receiving parameter packages from clients, the central server aggregates the global model. Since clients have different feature spaces, the aggregation algorithm only processes feature dimensions that appear in at least one client. For each global feature dimension indexed by $j \in \{0,1, \dots, D-1\}$, the server identifies all clients that use this feature (i.e., $j \in I_k$) and computes the global weight coefficient θ^j by averaging the local weight coefficients according to Eq. (3):

$$\theta^j = \frac{\sum_{k:j\in I_k} w_k . \theta_k^{(j)}}{\sum_{k:j\in I_k} w_k} \quad (3)$$

Where:

K: number of clients

D: total number of global features (global_feature_size)

S_k: number of training samples from client k

$S = \sum_{k=1}^{K} S_k$: total number of global samples

$w_k = \frac{S_k}{S}$: sample-based weight for client k

$I_k \subset \{0, \dots, D-1\}$: feature index set of client k

$\theta_k^{(j)}$: index feature coefficient j (global index) of client k, if $j \in I_k$

If no client has feature j, then:

$$\theta^j = 0$$

Additionally, the global bias coefficient b is calculated by the weighted average method: $b = \sum_{k=1}^{K} w_k . b_k$, where b_k is the bias coefficient of client k.

This aggregation method ensures that clients only train and send relevant features based on their device capabilities. The global model maintains accuracy by only aggregating dimensions that have meaningful information. It also ensures fairness among clients by weighting them according to their data size. As a result, the model demonstrates higher adaptability in scenarios where there is data and feature space heterogeneity among IoT devices.

The aggregation process is further clarified in Algorithm 3. First, the server initializes a global weight vector and a vector for storing the total weight at each feature dimension. Then, for each client, the algorithm iterates over the features that the client possesses and updates the weighted sum into the two vectors. After all clients have been processed, the global weight at each dimension is computed by dividing the weighted sum by the total weight. Finally, the global bias is calculated as the weighted average across all clients.

Algorithm 3: Federated Model Aggregation with Heterogeneous Feature Spaces

Input:

- Set of client models $\{(\theta_k, b_k)\}_{k=1}^{K}$
- Feature index sets $\{I_k\}_{k=1}^{K}$
- Sample sizes $\{S_k\}_{k=1}^{K}$
- Total feature dimension D

Output:

- Global model θ, global intercept b

1: Initialize $\theta \leftarrow 0 \in R^D$
2: Initialize $W \leftarrow 0 \in R^D$
3: Compute total samples: $S = \sum_{k=1}^{K} S_k$
4: For $k = 1$ to K do:
5: Compute weight $w_k \leftarrow \frac{S_k}{S}$
6: For each $j \in I_k$ do:
7: $\theta_j \leftarrow \theta_j + w_k . \theta_k^{(j)}$
8: $W_j \leftarrow W_j + w_k$
9: End for
10: End for
11: For $j = 1$ to D do:
12: If $W_j > 0$ then:
13: $\theta_j \leftarrow \frac{\theta_j}{W_j}$
14: Else:
15: $\theta_j \leftarrow 0$
16: End if
17: End for
18: Compute global bias: $b = \sum_{k=1}^{K} w_k . b_k$
19: Return (θ, b)

Weight update

After the aggregation stage, the server sends the global model (θ, b) to the clients. To address feature space heterogeneity, each client directly updates the feature dimensions in the local index set I_k. Dimensions outside of I_k and unrelated weights remain unchanged. The bias term is directly synchronized with the global value. The weight update process is outlined in Algorithm 4.

Algorithm 4: Global-to-Client Update under Heterogeneous Feature Spaces

Input:

- Global model (θ, b)
- Local feature index set I_k
- Previous local model $(\theta_k^{(t)}, b_k^{(t)})$

Output:

- Updated local model $(\theta_k^{(t+1)}, b_k^{(t+1)})$

1: Initialize $\theta_k^{(t+1)} \leftarrow \theta_k^{(t)}$
2: For each $j \in I_k$ do
3: $\theta_k^{(t+1,j)} \leftarrow \theta_j$
4: End for

5: For each $j \notin I_k$ do

6: $\theta_k^{(t+1,j)} \leftarrow \theta_k^{(t,j)}$

7: End for

8: Update bias: $b_k^{t+1} \leftarrow b$

9: Return $\left(\theta_k^{(t+1)}, b_k^{(t+1)}\right)$

IV. EXPERIMENTS

A. Experimental Environment

The experiments were conducted on the same virtualization system running the Windows 10 64-bit operating system, equipped with an Intel Core i7-7820HQ processor (2.9 GHz) and 16GB of RAM. In this study, we simulate the Federated Learning environment with three clients, deployed as three Ubuntu virtual machines running on VMware Workstation. These virtual machines represent three IoT devices with different computing capabilities, simulating the heterogeneous nature of real-world IoT ecosystems. Our physical machine serves as the central server coordinating the entire federated training process.

Each client is configured with different computing resources, ranging from 1GB to 2GB of RAM and 1 to 2 vCPUs, to accurately replicate the processing discrepancies between IoT devices. Data is independently distributed to each client with no overlap, ensuring the distributed nature of the task. Specifically, to simulate the differences in feature space between the devices, each client is assigned a different number of features. This results in local models that are heterogeneous in terms of parameter numbers, posing a significant challenge for the model aggregation process in Federated Learning.

B. Dataset

The dataset used for testing consists of selected samples from the C500-IoT dataset collected by Phu [21] and Trung [22] with a total of 8,786 ELF file samples, including 4,393 benign files and 4,393 malicious files.

C. Experimental results and evaluation

To thoroughly evaluate the suitability of the proposed method for resource-constrained IoT devices, two key practical evaluation metrics are reported in the results tables. Model size (KB) refers to the total storage memory required for the model weights and parameter checkpoints on the client device. Lower values indicate a smaller storage footprint, making the model more suitable for devices with limited memory. Inference time (ms) refers to the average execution time required to classify a single binary file sample. Lower inference times enable efficient real-time detection without overloading the device CPU.

The experimental data is split with 80% allocated for training and 20% for testing. However, in Federated Learning, the 80% training data is partitioned and distributed to each client, ensuring that each device only processes its own local data. The 20% testing data remains unchanged to evaluate the global model after being aggregated through several rounds of Federated Learning. Model parameter optimization is performed through grid search combined with cross-validation to ensure the optimal configuration for each machine learning algorithm. The key parameters for the machine learning algorithms are presented in Table I.

TABLE I. KEY PARAMETERS OF MACHINE LEARNING ALGORITHMS

Machine Learning Algorithm	**Parameters**	**Values**
Support Vector Machine	Alpha	0.0001
	learning_rate	‘optimal’
	max_iter	3000
	penalty	‘l2’
	random_state	42
	Tol	0.0001
Random Forest	bootstrap	‘False’
	max_depth	‘None’
	max_features	‘log2’
	min_samples_leaf	1

	min_samples_split	2
	n_estimators	200
	random_state	42
Naive Bayes	Alpha	0.00001
	fit_prior	'True'

- Feature selection method evaluation

In the first scenario, the study focuses on evaluating the effectiveness of two popular feature representation methods in opcode processing: TF-IDF and Count Vectorizer, using the widely known SVM machine learning algorithm as the classification model. The objective is to identify the optimal feature representation method for opcode sequences.

The experimental results, presented in Table II, show that the malicious code detection model using the TF-IDF method significantly outperforms the model using the Count Vectorizer method across all evaluation metrics. Specifically, TF-IDF achieves the highest accuracy of 95.6% and an F1-score of 95.8%, while Count Vectorizer reaches a maximum accuracy of only 87.2% and an F1-score of 85.5%. This difference stems from the weighting mechanism of TF-IDF, where the Inverse Document Frequency (IDF) helps reduce the impact of common opcodes that appear in most samples, while highlighting rare opcodes that have a high discriminative ability between benign and malicious code. As a result, the model becomes more focused on important features, minimizing noise caused by the dominance of frequent opcodes.

From this result, TF-IDF is selected as the primary feature vectorization method for the proposed IoT malware detection model. More importantly, in Federated Learning, TF-IDF also facilitates better synchronization between clients with different opcode counts.

- Evaluation of effective Machine Learning algorithms

The next scenario evaluates the performance of three popular machine learning algorithms: Support Vector Machine (SVM), Random Forest (RF), and Naive Bayes (NB). The aim is to select the most suitable algorithm for constructing the detection model, which will later combine with the Federated Learning method, and to analyze the impact of the number of features on the detection quality of the machine learning model.

The experimental results, shown in Table III, indicate that Random Forest achieves the highest performance with an accuracy of 99.2% and an F1-score of 99.3%. However, due to its ensemble nature requiring the construction of a large number of decision trees (*n_estimators = 200*), Random Forest incurs significant computational overhead during both training and inference, leading to increased execution time as observed in Table III, which makes it unsuitable for the Federated Learning environment. SVM, on the other hand, provides a balance between accuracy and complexity, achieving an accuracy of 95.6% and an F1-score of 95.8%, with a detection time of 1.23 ms. Notably, the compact parameter structure of SVM makes it easier to aggregate and synchronize in the decentralized Federated Learning environment, thus it is selected as the core algorithm for the proposed system. Conversely, Naive Bayes performs the worst with a maximum accuracy of only 79.8% due to the assumption of feature independence, which is not appropriate for opcode data, where many complex dependencies exist.

From these experiments, the proposed detection model utilizes the TF-IDF feature extraction method and the SVM machine learning algorithm, which will be integrated with the proposed Federated Learning method.

D. Comparison of the proposed IoT malware detection model with other federated learning models

To compare the proposed IoT malware detection model, we compare it with several existing models using the Federated Learning method on the same dataset and experimental environment as in this study. The comparison results are shown in Table IV.

TABLE II. COMPARISON OF FEATURE VECTORIZATION METHODS

Model	25 opcodes						30 opcodes						35 opcodes					
	Acc (%)	*Precision (%)*	*Recall (%)*	*F1-Score (%)*	*Size (KB)*	*Time (ms)*	*Acc (%)*	*Precision (%)*	*Recall (%)*	*F1-Score (%)*	*Size (KB)*	*Time (ms)*	*Acc (%)*	*Precision (%)*	*Recall (%)*	*F1-Score (%)*	*Size (KB)*	*Time (ms)*
TFIDF-SVM	**95.1**	**95.8**	**94.3**	**95.1**	**0.2**	**1.45**	**94.9**	**93.6**	**96.4**	**94.9**	**0.24**	**1.12**	**95.6**	**95.8**	**95.5**	**95.8**	**0.28**	**1.23**
Count-SVM	84.1	83.4	83.2	81.0	0.2	1.6	85.0	83.9	82.2	83.0	0.24	1.63	87.2	82.2	85.5	83.8	0.28	1.71

TABLE III. EVALUATION OF MACHINE LEARNING ALGORITHMS

Model	25 opcodes						30 opcodes						35 opcodes					
	Acc (%)	*Precision (%)*	*Recall (%)*	*F1-Score (%)*	*Size (KB)*	*Time (ms)*	*Acc (%)*	*Precision (%)*	*Recall (%)*	*F1-Score (%)*	*Size (KB)*	*Time (ms)*	*Acc (%)*	*Precision (%)*	*Recall (%)*	*F1-Score (%)*	*Size (KB)*	*Time (ms)*
TFIDF-SVM	95.1	95.8	94.3	95.1	0.2	1.45	94.9	93.6	96.4	94.9	0.24	1.1	95.6	95.8	95.5	95.8	0.28	1.23
TFIDF-RF	**98.7**	**99.1**	**98.8**	**98.4**	**0.05**	66.6	**99.1**	**98.9**	**98.4**	**98.6**	**0.05**	68.1	**99.2**	**99.4**	**99.4**	**99.3**	**0.05**	68.4
TFIDF-NB	79.1	71.5	96.8	82.2	**0.05**	**0.52**	79.3	71.7	96.8	82.4	**0.05**	**0.55**	79.8	72.1	97.1	82.8	**0.05**	**0.55**

TABLE IV. COMPARISON WITH OTHER FEDERATED LEARNING MODELS

Model	**Machine Learning Algorithm**	**Acc** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
Lin and Huang [19]	SVM	90.9	90.1	91.0	90.6
Hsu et al. [18]	SVM	89.7	89.2	90.1	89.6
Jiang et al. [16]	FedHGCDroid	91.6	91.1	92.1	91.6
Taheri et al. [20]	FedGAN-BK	90.1	91.4	89.4	90.4
Proposed Model	FL-SVM	**94.9**	**93.6**	**96.4**	**94.9**

To ensure a fair and objective comparison, all baseline models were not evaluated based on their originally reported results but were consistently re-implemented within the same experimental framework. Specifically, the compared methods, including [16, 18-20], were reproduced using their publicly available source code and retrained on the same C500-IoT dataset utilized in this study. In addition, key hyperparameters such as the number of communication rounds, client participation ratio, and learning rate were carefully tuned for each model to achieve optimal performance under identical conditions. This unified experimental setup ensures that the performance differences observed are attributable to the proposed method rather than discrepancies in data, configuration, or training procedures.

The proposed IoT malware detection model using the SVM algorithm combined with Federated Learning outperforms the other models in terms of accuracy. The proposed IoT malware detection model using Federated Learning maintains stable performance even under conditions where clients have differing feature spaces - a challenge that previous studies have not addressed effectively. Additionally, this approach allows for flexible system expansion by adding new clients with diverse hardware configurations without changing the global model structure, enhancing portability and practical deployment in decentralized IoT environments.

V. CONCLUSION

This study proposes a malware detection method for IoT devices based on a Federated Learning model combined with opcode sequence analysis. The model is trained on multiple clients, each with a different number of features depending on the device's processing capability. The results show that even in a distributed and heterogeneous feature environment, using TF-IDF and the feature selection method, combined with the proposed Federated Learning model

aggregation mechanism, achieves high accuracy in detecting IoT malware.

In the future, this model could be extended in several ways: integrating dynamic features alongside static opcode features to improve detection of malware using advanced evasion techniques such as obfuscation, encryption, or polymorphism; researching more advanced model aggregation mechanisms to better handle feature number differences between devices, particularly when the number of clients increases; or integrating lightweight deep learning models in Federated Learning to improve accuracy while meeting the resource requirements of IoT devices; or incorporate experimental methods to prove data transmission bandwidth minimization in real networks.

REFERENCES

[1] "Enabling-IOT.pdf", Access time:18/12/2025, *https://cdn.ihs.com/www/pdf/enabling-IOT.pdf*

[2] "TL-WR841N", Access time: 17/12/2025, https://volatilesystems.org/hardware/tl-wr841n.html.

[3] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A Vishwanath, V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics", *IEEE Trans. on Mobile Comput.,* vol. 18, no. 8, pp. 1745–1759, 2019. DOI: 10.1109/TMC.2018.2866249.

[4] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, ... & Y. Zhou, "Understanding the Mirai Botnet", *in Proc. 26th USENIX Security Symp. (USENIX Security 2017),* Vancouver, BC, Canada, pp. 1093–1110, Aug. 16–18, 2017.

[5] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, & S. Fahl, "Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security", *in 2017 IEEE Symposium on Security and Privacy (SP*), San Jose, CA, USA: IEEE, pp. 121–136, 2017. DOI: 10.1109/SP.2017.31.

[6] C.W. Tien, S.W. Chen, T. Ban, and S.Y. Kuo, "Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features", *Digital Threats,* vol. 1, no. 1, pp. 5:1-5:19, 2020. DOI: 10.1145/3378448.

[7] N. N. Toan, L. T. Dung, D. Q. Thang, "Static Feature Selection for IoT Malware Detection", *Journal of Science and Technology on Information security*, vol. 1, no. 15, pp. 74 - 84, 2022. DOI: 10.54654/isj.v1i15.844.

[8] L. T. Dung, N. N. Toan, T. N. Phu, "CAIMP: Cross-Architecture IoT Malware Detection and Prediction Based On Static Feature", *The Computer Journal,* vol. 67, no. 9, pp. 2763 - 2776, 2024. DOI:10.1093/comjnl/bxae042

[9] J. Ramamoorthy, K. Gupta, R. C. Kafle, N. K. Shashidhar, and C. Varol, "A Novel Static Analysis Approach Using System Calls for Linux IoT Malware Detection", *Computer Science and Mathematics*, 2024*,* DOI: 10.20944/preprints202407.0268.v1.

[10] J. Ramamoorthy, K. Gupta, N. K. Shashidhar, and C. Varol, "Linux IoT Malware Variant Classification Using Binary Lifting and Opcode Entropy", *Electronics,* vol. 13, no. 12, pp. 2381, 2024. DOI: 10.3390/electronics13122381.

[11] T. A. Tu, D. C. Thanh, T. D. Su, "Amplified Gradient Inversion Attacks on Federated Learning Frameworks", *Journal of Science and Technology on Information Security,* vol. 3, no. 23, pp. 15–26, 2024. DOI: 10.54654/isj.v3i23.1066

[12] V. Rey, P. M. Sánchez, A. Huertas Celdrán, and G. Bovet, "Federated learning for malware detection in IoT devices", *Computer Networks,* vol. 204, pp. 108693, 2022. DOI: 10.1016/j.comnet.2021.108693.

[13] Z. Çıplak, K. Yıldız, and Ş. Altınkaya, "FEDetect: A Federated Learning-Based Malware Detection and Classification Using Deep Neural Network Algorithms", *Arab J Sci Eng*, 2025. DOI: 10.1007/s13369-025-10043-x.

[14] M. Nobakht, R. Javidan, and A. Pourebrahimi, "SIM-FED: Secure IoT malware detection model with federated learning", *Computers and Electrical Engineering,* vol. 116*,* pp. 109-139, 2024. DOI: 10.1016/j.compeleceng.2024.109139.

[15] M. Asiri, M. A. Khemakhem, R. M. Alhebshi, B. S. Alsulami, and F. E. Eassa, "RPFL: A Reliable and Privacy-Preserving Framework for Federated Learning-Based IoT Malware Detection", *Electronics,* vol. 14, no. 6, pp. 1089, 2025. DOI: 10.3390/electronics14061089.

[16] C. Jiang, K. Yin, C. Xia, and W. Huang, "FedHGCDroid: An Adaptive Multi-Dimensional Federated Learning for Privacy-Preserving Android Malware Classification", *Entropy,* vol. 24, no. 7, pp. 919, 2022. DOI: 10.3390/e24070919.

[17] K. Chen, W. Zhang, Z. Liu, and B. Mi, "Leveraging Federated Learning for Malware Classification: A Heterogeneous Integration

Approach", *Electronics,* vol. 14, no. 5, pp. 915, 2025. DOI: 10.3390/electronics14050915.

[18] R. H. Hsu, Y. C. Wang, C. I. Fan, B. Sun, T. Ban, T. Takahashi, ... & S. W. Kao, "A Privacy-Preserving Federated Learning System for Android Malware Detection Based on Edge Computing", *in 2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, pp. 128-136, 2020. DOI: 10.1109/AsiaJCIS50894.2020.00031.

[19] K. Y. Lin and W. R. Huang, "Using Federated Learning on Malware Classification", *in 2020 22nd International Conference on Advanced Communication Technology (ICACT),* pp. 585–589, 2020. DOI: 10.23919/ICACT48636.2020.9061261.

[20] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, "Fed-IIoT: A Robust Federated Malware Detection Architecture in Industrial IoT", *IEEE Transactions on Industrial Informatics,* vol. 17, no. 12, pp. 8442–8452, 2021. DOI:10.1109/TII.2020.3043458.

[21] T. N. Phu, H. D. Kien, N. Q. Dung, N. D. Tho, "A Novel Framework to Classify Malware in MIPS Architecture-Based IoT Devices", *Hindawi Security and Communication Networks Volume 2019*, Article ID 4073940, 2019. DOI: 0.1155/2019/4073940

[22] N. H. Trung, "Doctoral thesis Research to propose PSI graph characteristics in detecting Botnet malware on IoT devices", PhD Thesis, Academy of Science and Technology, Vietnam Academy of Sciences, 2021.

ABOUT THE AUTHOR

Nguyen Ngoc Toan

Workplace: People's Security Academy, Vietnam

Email: ngoctoan.hvan@gmail.com

Education: Received Bachelor's degree in 2015 from PSA, PhD degree in 2025 in Information Security from Academy of Cryptography Techniques.

Recent research direction: IoT malware detection, AI, machine learning for information security.

Tên tác giả: **Nguyễn Ngọc Toàn**

Cơ quan công tác: Học viện An ninh nhân dân, Việt Nam

Email: ngoctoan.hvan@gmail.com

Quá trình đào tạo: Tốt nghiệp đại học tại Học viện An ninh nhân dân năm 2015 và nhận bằng tiến sĩ ngành An toàn thông tin tại Học viện Kỹ thuật mật mã năm 2025.

Hướng nghiên cứu hiện nay: Phát hiện mã độc, Trí tuệ nhân tạo, Học máy trong an trong an ninh mạng.

Nguyen Manh Tuan

Workplace: Ministry of Public Security, Vietnam

Email:

tuannguyentc2002@gmail.com

Education: Received Bachelor's degree in 2026 from PSA.

Recent research direction: Malware detection, machine learning applications.

Tên tác giả: **Nguyễn Mạnh Tuấn**

Cơ quan công tác: Bộ Công an, Vietnam

Email: tuannguyentc2002@gmail.com

Quá trình đào tạo: Tốt nghiệp đại học tại Học viện An ninh nhân dân năm 2026.

Hướng nghiên cứu hiện nay: Phát hiện mã độc, Các ứng dụng học máy.