IoT-FedMalDetect: Federated Learning based Malware Detection for IoT Edge Devices

Heetkumar Patel, Suresh Kumar Amalapuram, **Saurabh Kumar**, Bheemarjuna Reddy Tamma

Indian Institute of Technology Hyderabad

(10th September 2025)

@ IEEE CNS 2025

Acknowledgement

- ISEA Phase-III project, titled "Security in Distributed Wireless Networks", funded by MeitY, Government of India
- DST-FIST Grant

Motivation

- Rapid proliferation of IoT devices.
  - Across homes, industries, and critical infrastructure.
- Rise of IoT malwares.
  - e.g., Mirai, Bashlite botnets causing large-scale disruptions.
- Traditional signature-based detection systems are ineffective against novel malware.
- Privacy concerns prevent centralized data collection from IoT devices.
- We need a solution that is
  - Scalable
  - Efficient
  - Privacy-preserving

Problem Statement

Objective: Enable efficient malware detection in IoT edge devices.

Challenges:

- Resource-constrained devices (limited memory, CPU)
- Lack of labeled data for supervised learning
- Privacy concerns prevent data sharing
- Non-IID and highly imbalanced data distribution across devices.

Challenges in existing work

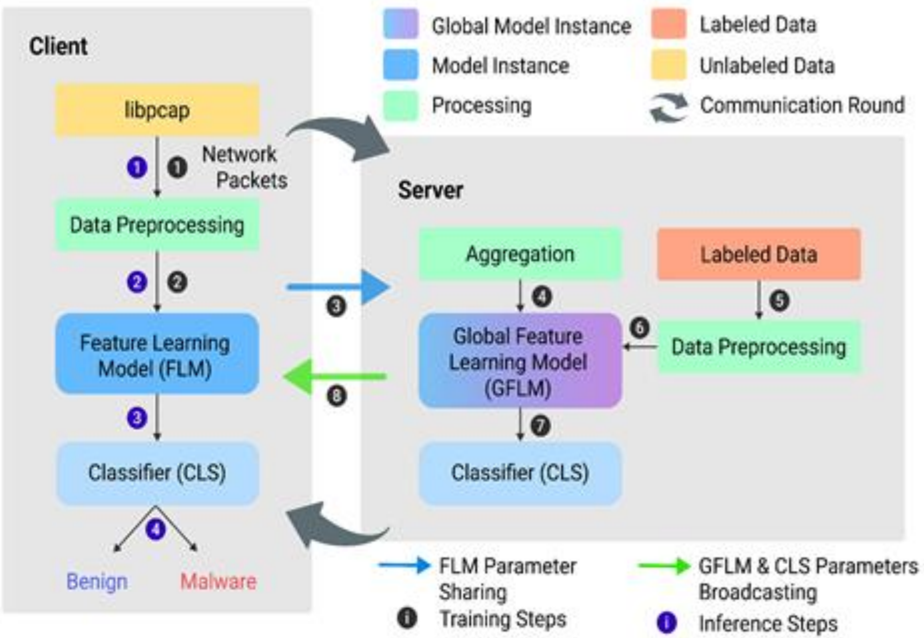
Federated Semi-supervised / Unsupervised approach:

- Pseudo-labeling methods[1]: rely on confidence scores.
- Autoencoder-based approaches[2]: use fixed reconstruction thresholds.
- Centroid-based classifiers[3]: depend on a fixed distance from centroid.

Common issue: Lack robustness to real-world variability, because of fixed separation value.

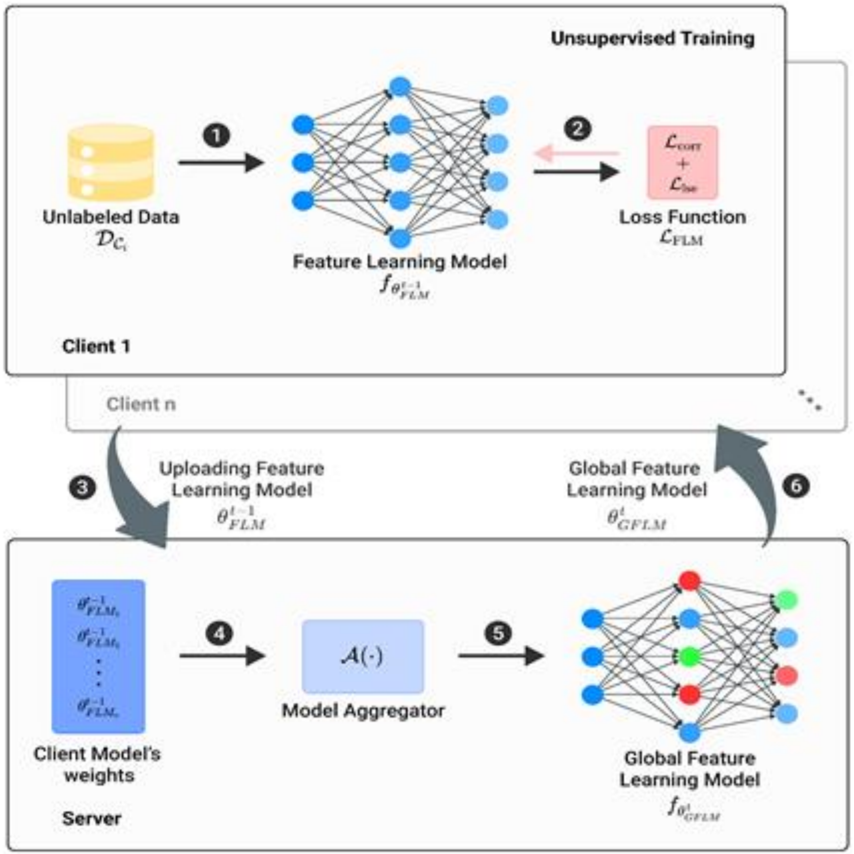
1. X. Pei et. al. “A knowledge transfer-based semi-supervised federated learning for IoT malware detection,” DTSC 2022
2. Y. Meidan et. al “N-baiot—network-based detection of iot botnet attacks using deep autoencoders,” IEEE Pervasive Computing 2018
3. V. T. Nguyen, R. Beuran “FedMSE: Semi-supervised federated learning approach for IoT network intrusion detection,” Computers & Security 2025

System Overview



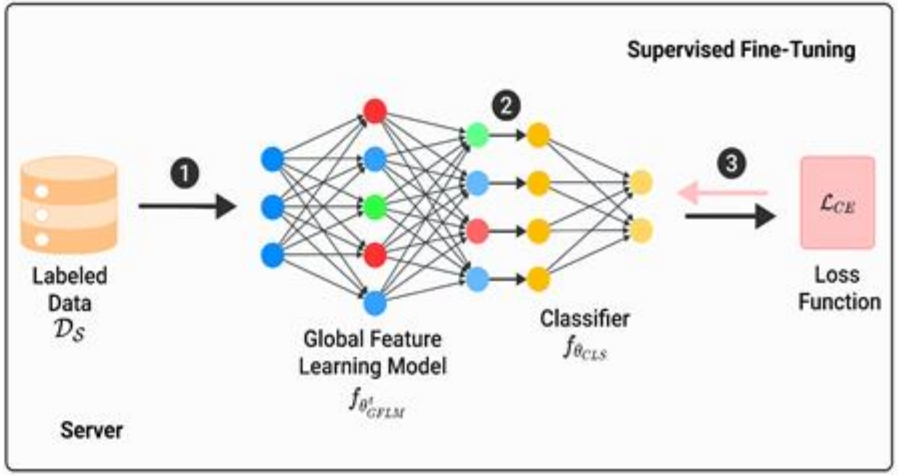
- Client (IoT Devices)
  - Capture network traffic using Scapy
  - Preprocess raw traffic: extract features
  - Locally train Feature Learning Model (FLM)
- Central Server
  - Aggregates FLM parameters from multiple clients using **FedDyn**
  - Fine-tunes a global model by adding a classifier head trained on public labeled data
  - Distributes final detection model back to IoT clients.

Unsupervised FLM training (Client)



- Goal: Learn meaningful feature representations without labels.
- Loss Functions:
  - Correlation Loss (L_{corr}): Minimizes redundancy in learned features.
  - Latent Space Equalization Loss (L_{lse}): Ensures balanced feature utilization and prevents feature collapse.
- Clients send only model parameters (no raw data) to server
- Server aggregates models using **FedDyn**, which reduces client drift caused by non-IID data.

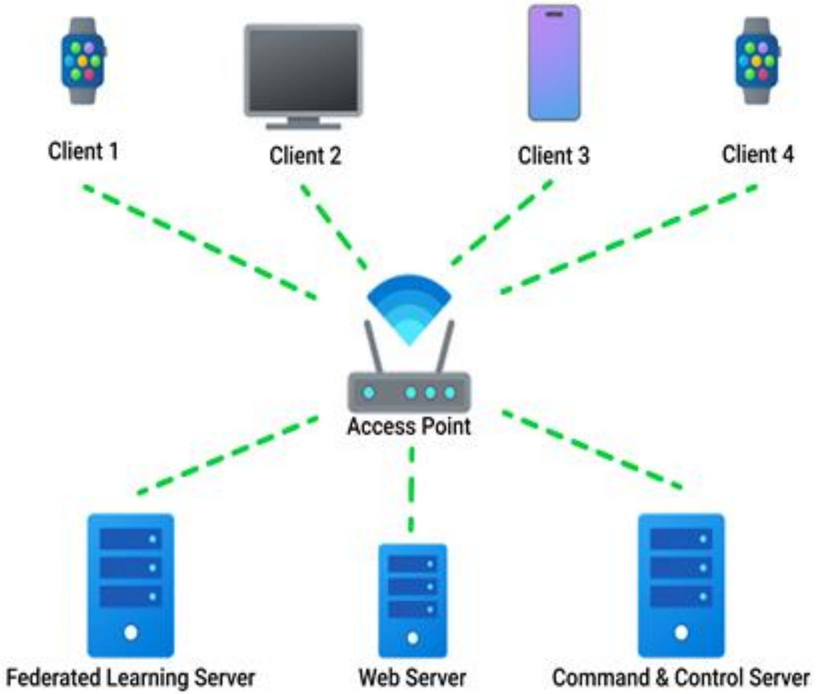
Supervised Fine-tuning (Server)



- The server attaches a classifier head to the aggregated global FLM
- Fine-tunes the combined model using limited labeled malware samples
- The final model is broadcast back to IoT clients for deployment.
- Advantages:
  - Eliminates need for labeled data at clients
  - Allows continuous adaptation to new threats.

Experimental Setup

- IoT Testbed Simulation:
  - Raspberry Pi devices as clients, one as C&C server, another as central server
  - Captured traffic includes both benign activities and botnet (Mirai, Bashlite) attacks
- Local Simulation:
  - High-performance server simulating multiple clients
  - Datasets: N-BaIoT, IoT-23, RaDaR
- Feature extraction: 115 statistical features over multiple time windows.



IoT Testbed

Data Distribution and Preprocessing

- Data split: 70% Train, 10% Validation, 20% Test
- Dirichlet partitioning simulates non-IID distribution
- Features normalized using Standard Scaler
- Data split across multiple clients to simulate real-world heterogeneity.

Performance Metrics

- **ROC-AUC:** Measure of true positive rate vs false positive rate
- **PR-AUC:** Measure of precision vs recall
- Resource usage metrics:
  - CPU utilization (%)
  - Memory usage (MB)
  - FLOPs and model size (parameters)

Result [1/6]: Validation on IoT Testbed

Model	Device	Bashlite	Mirai	Training Samples (Benign)	Training Samples (Malware)	Training Time (sec)	Memory Consumption (MB)	CPU Consumption (out of 400%)	ROC-AUC	PR-AUC
Our	Client-1	✓	✓	49548	30090	160	599.43	212	0.9770	0.9657
	Client-2	✗	✓	13113	17378	94	249	212	0.9312	0.9590
	Client-3	✓	✗	39100	12180	140	403	211	0.8743	0.7809
	Client-4	✗	✗	45627	-	132	367	211	0.9211	0.9510
SAE-CEN	Client-1	✓	✓	49548	-	233	498	224	0.9994	0.9929
	Client-2	✗	✓	13113	-	70	137	225	0.9712	0.9672
	Client-3	✓	✗	39100	-	176	499	224	0.9938	0.9777
	Client-4	✗	✗	45627	-	211	458	224	0.9397	0.8942

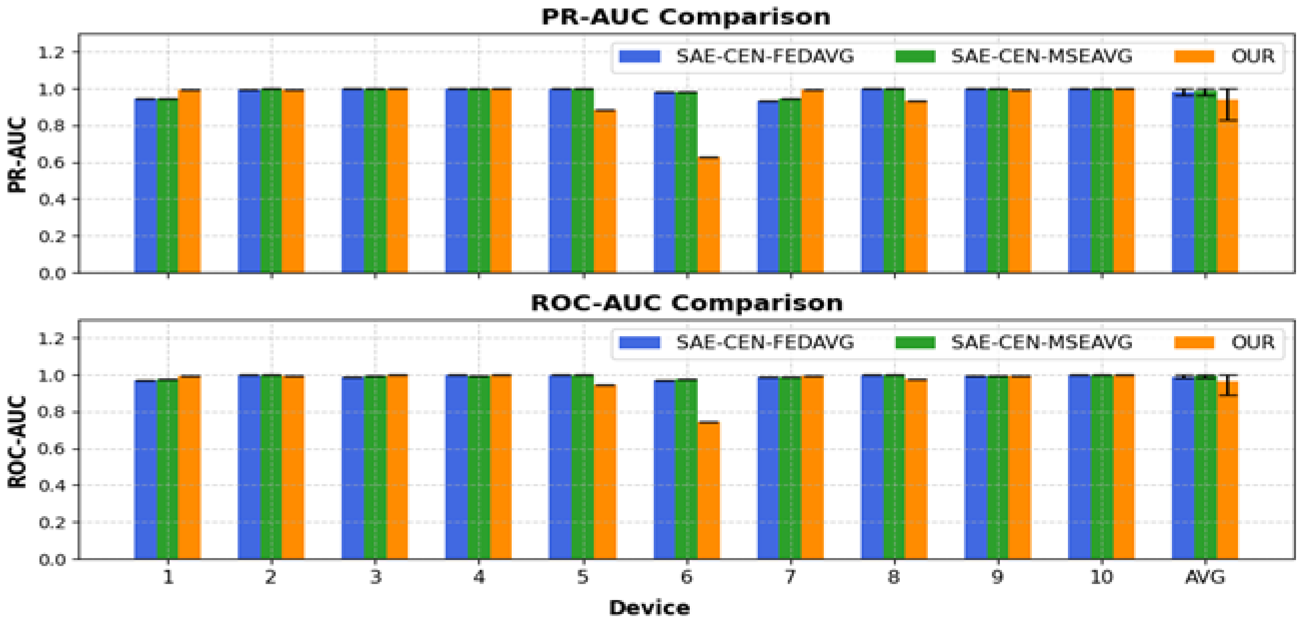
Result [1/6]: Validation on IoT Testbed

Model	Device	Bashlite	Mirai	Training Samples (Benign)	Training Samples (Malware)	Training Time (sec)	Memory Consumption (MB)	CPU Consumption (out of 400%)	ROC-AUC	PR-AUC
Our	Client-1	✓	✓	49548	30090	160	599.43	212	0.9770	0.9657
	Client-2	✗	✓	13113	17378	94	249	212	0.9312	0.9590
	Client-3	✓	✗	39100	12180	140	403	211	0.8743	0.7809
	Client-4	✗	✗	45627	-	132	367	211	0.9211	0.9510
SAE-CEN	Client-1	✓	✓	49548	-	233	498	224	0.9994	0.9929
	Client-2	✗	✓	13113	-	70	137	225	0.9712	0.9672
	Client-3	✓	✗	39100	-	176	499	224	0.9938	0.9777
	Client-4	✗	✗	45627	-	211	458	224	0.9397	0.8942

IoT-FedMalDetect achieves comparable detection performance without knowledge of the labels of any class at client

Model trains ~1.6X faster, and uses ~1.2X less memory with some overhead on server

Result [2/6]: Local Simulation on N-BaIoT Dataset with 10 Clients



Our method achieves competitive performance, enabling effective learning without compromising privacy or relying on centralized data, as in baseline approaches

Result [3/6]: Performance Validation

Performance on Centralized vs Federated training setup on N-BaIoT dataset with 10 Clients

Metric	Centralized	FL (avg)
ROC-AUC	0.9988	0.9485
PR-AUC	0.9988	0.9177

Performance on Various Malware Dataset with 10 Clients

Metric	N-BaIoT	IoT-23	RaDaR
Avg. ROC-AUC	0.9485	0.8549	0.7350
Avg. PR-AUC	0.9177	0.9311	0.9506

Federated Learning shows strong detection with a minor drop compared to centralized training due to diverse, distributed client data hindering uniform pattern learning

Our model generalizes well, with high PR-AUC even on RaDaR, though lower ROC-AUC suggests some sensitivity to class imbalance

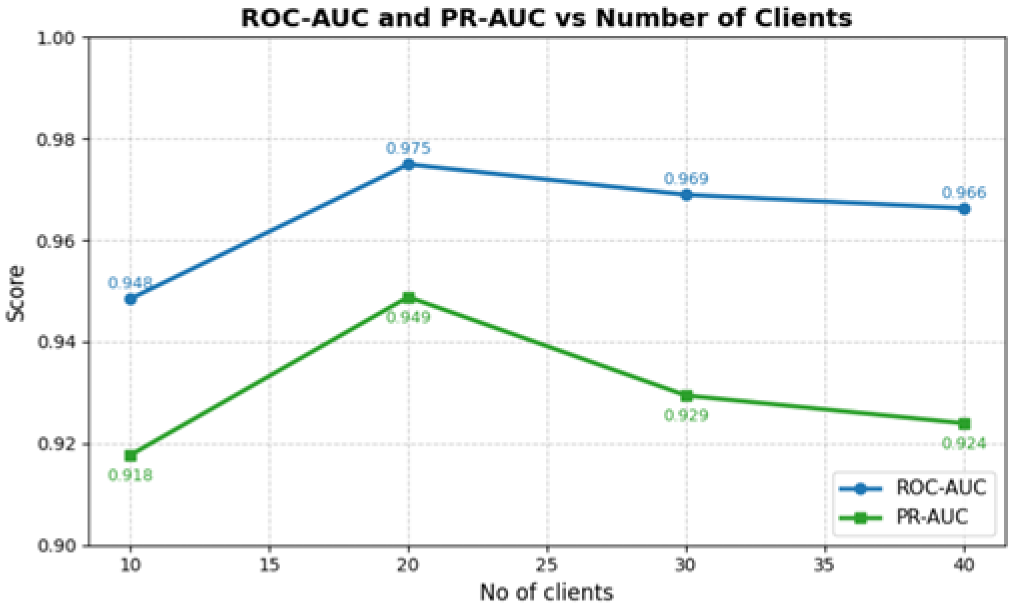
Result [4/6]: Computation Efficiency

Against SAE-CEN (FedMSE variants)

Metrics	SAE-CEN		Ours	
	Client	Server	Client	Server
FLOPs (KFLOPS)	303.6	-	**11.65**	41.538
MACs (KMACs)	151.2	-	**5.75**	20.47
Params (K)	12.826	-	**9.164**	24.334

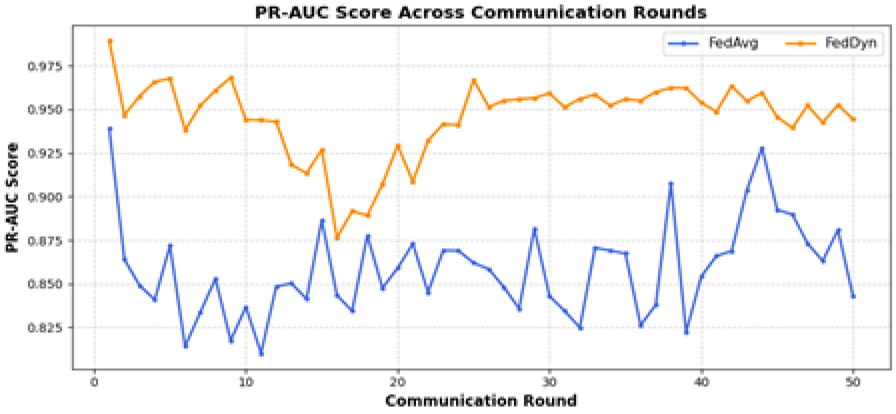
Our method drastically reduces client-side computation with overhead at server-side, making it highly suitable for IoT devices

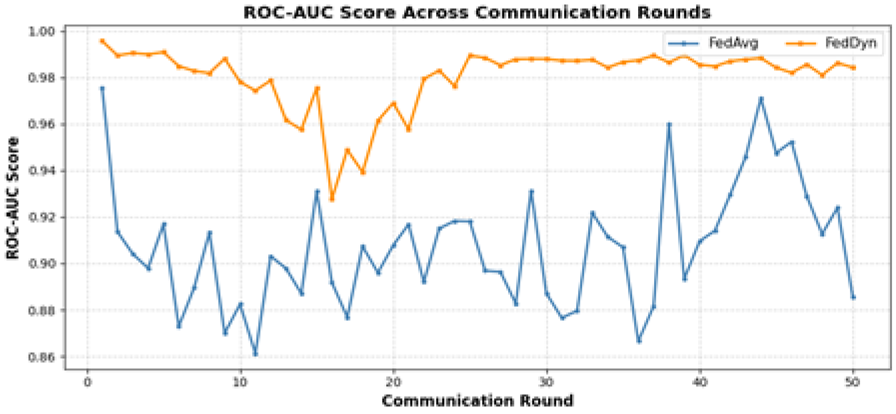
Result [5/6]: With Varying #Clients



Detection performance fluctuates but remains high and robust, reflecting the FL trade-off where more clients increase heterogeneity, noise, and variance

Result [6/6]: Aggregation Methods (FedAvg & FedDyn)





FedDyn shows better stability and performance compared to FedAvg across communication rounds, which makes it more reliable for FL in non-IID settings

FedAvg suffers from high fluctuations, indicating sensitivity to client updates and data heterogeneity

Conclusion

Proposed a semi-supervised federated learning framework for IoT malware detection

Unsupervised client-side training - no need for labeled data on devices

Achieves comparable detection performance to baseline models

Lightweight training - suitable for resource-constrained IoT devices

Provides a scalable, efficient, and privacy-preserving IoT security solution

Thank You