Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080940

ARTICLE

Privacy-Preserving Federated Malware Detection Using Memory and Behavioral Features

Ammar Odeh*, Osama Alhaj Hassan and Anas Abu Taleb

Department of Computer Science, Princess Sumaya University of Technology, Amman, Jordan

*Corresponding Author: Ammar Odeh. Email: a.odeh@psut.edu.jo

Received: 19 February 2026; Accepted: 13 April 2026; Published: 15 June 2026

ABSTRACT: The rapid growth of sophisticated malware and the increasing diversity of computing environments have exposed critical limitations in traditional centralized malware detection systems, particularly in data privacy, scalability, and adaptability. This study proposes a privacy-preserving, collaborative malware-detection framework that leverages federated learning to improve detection accuracy while keeping sensitive data local to participating devices. The objective is to address emerging malware threats by combining behavioral and memory-based analysis within a decentralized learning paradigm. The proposed framework employs federated learning to train a global malware detection model without transferring raw data. Each client locally extracts discriminative features derived from system behavior and memory artifacts, including process activity patterns, memory access characteristics, and runtime indicators. Local deep learning models are trained independently, and only model parameters are shared with a central aggregator, which constructs an optimized global model through iterative parameter aggregation. This approach significantly reduces privacy risks and communication overhead compared to centralized training. Experimental evaluations on benchmark malware datasets demonstrate that the proposed federated approach achieves detection performance comparable to, and in some cases exceeding, that of centralized deep learning models. The results indicate improved robustness against previously unseen malware variants, with high detection accuracy and reduced false positive rates. Furthermore, privacy is preserved throughout the learning process, making the framework suitable for real-world distributed, resource-constrained environments. The findings confirm that federated learning, combined with memory and behavioral feature analysis, provides an effective, privacy-aware solution for modern malware detection. This work contributes to recent advances in cybersecurity by offering a scalable, secure, and practical detection framework that can be deployed across distributed systems, including enterprise networks and edge computing environments.

KEYWORDS: Malware detection; federated learning; privacy-preserving machine learning; memory forensics; behavioral analysis; deep learning; distributed cybersecurity; threat intelligence

1 Introduction

The contemporary digital ecosystem faces an unprecedented escalation in cyber threats, with malware attacks growing exponentially in both volume and sophistication. Recent years have witnessed the emergence of advanced malware variants, including ransomware that encrypts critical infrastructure, spyware capable of evading behavioral analysis, distributed botnets orchestrating large-scale DDoS attacks, fileless malware residing solely in memory, and zero-day exploits that compromise systems before patches become available [1,2]. This malware evolution is further compounded by the dramatic expansion of attack surfaces across heterogeneous computing environments—encompassing cloud infrastructures, Internet of Things

(IoT) devices, mobile platforms, Industrial IoT (IIoT) systems, and edge computing nodes. Each of these domains introduces unique vulnerabilities and deployment constraints that challenge traditional security paradigms [3,4].

Traditional security mechanisms, particularly signature-based antivirus solutions and rule-based intrusion detection systems (IDS), have proven increasingly inadequate against modern threats [5,6]. These conventional approaches rely on predefined patterns and known attack signatures, rendering them fundamentally reactive rather than proactive [7]. The core challenge lies not merely in the volume of malware. Still, in its adaptive nature, attackers continuously evolve their techniques through polymorphic code generation, metamorphic transformations, and obfuscation strategies that systematically evade signature-based detection. This cat-and-mouse dynamic necessitates a paradigm shift toward intelligent, adaptive security frameworks capable of identifying previously unseen threats [8,9].

The inadequacy of traditional detection methodologies stems from several fundamental limitations [10]. Signature-based detection, while effective against known threats, performs poorly against polymorphic and metamorphic malware that dynamically alter their code structure while preserving malicious functionality. Each code transformation generates a unique signature, rendering database-dependent detection approaches obsolete upon the malware's first mutation [11].

Heuristic and rule-based systems attempt to address this limitation by identifying suspicious behaviors or patterns indicative of malicious activity. However, these approaches suffer from high false positive rates, as legitimate software occasionally exhibits behaviors that trigger heuristic rules. The rigid nature of predefined rules also limits their adaptability to novel attack vectors [12].

Hybrid code analysis approaches address critical limitations in traditional JavaScript optimization techniques, particularly in handling redundancy within complex and dynamic web applications. Relying solely on static analysis often fails to capture runtime behaviors such as dynamic function calls and event-driven execution, while dynamic analysis alone suffers from incomplete coverage and potential false positives. By integrating both methods, hybrid analysis enables more accurate identification of unused or redundant code, improving overall code efficiency and maintainability. This approach not only reduces application size and enhances loading performance but also supports scalability in large, modular systems where traditional optimization techniques struggle to provide comprehensive results [13].

Perhaps most critically, traditional approaches struggle with zero-day exploits and stealthy malware designed to operate beneath detection thresholds. Advanced persistent threats (APTs) employ sophisticated evasion techniques, including sandbox detection, delayed execution, and minimal system footprints, that evade both signature- and behavior-based detection mechanisms. These limitations collectively underscore the urgent need for advanced, intelligent detection techniques [14].

1.1 The Role of Machine Learning and Deep Learning

The cybersecurity community has increasingly turned to machine learning (ML) and deep learning (DL) approaches to address the shortcomings of traditional detection systems. ML-based malware detection leverages algorithms such as Support Vector Machines (SVMs), Random Forests (RFs), Naive Bayes (NB), and K-Nearest Neighbors (KNNs) to classify malware based on extracted features. These approaches demonstrate improved generalization capabilities and can identify previously unseen malware variants using learned patterns rather than explicit signatures [15].

Deep learning architectures have further advanced the field through automatic feature extraction and hierarchical representation learning:

- Convolutional Neural Networks (CNNs) excel in static analysis, processing malware binaries as grayscale images or analyzing opcode sequences to identify structural patterns indicative of malicious code.
- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks effectively model sequential data such as API call sequences, system logs, and dynamic execution traces, capturing temporal dependencies that characterize malicious behavior.
- Autoencoders enable unsupervised anomaly detection by learning compressed representations of benign behavior and flagging deviations as potential threats.

The advantages of ML/DL approaches are compelling: automatic feature learning eliminates manual feature engineering; improved generalization enables detection of novel malware families; and adaptive learning allows models to evolve with emerging threats. However, these approaches introduce new challenges, including data imbalance (benign samples vastly outnumber malicious ones), model explainability concerns that hinder trust and debugging, model drift as malware evolves beyond the training distribution, and computational costs that may be prohibitive in resource-constrained environments [7].

1.2 Emerging Trends and Recent Advances

Recent research has explored several promising directions that address limitations of both traditional and conventional ML-based approaches. Federated learning frameworks enable privacy-preserving malware detection by training models across distributed devices without centralizing raw data, addressing both privacy and scalability concerns. Edge-based and distributed detection architectures process data locally, reducing latency and bandwidth requirements while enabling real-time threat response.

Hybrid analysis frameworks combining static and dynamic analysis techniques leverage complementary strengths—static analysis provides comprehensive code coverage without execution overhead. In contrast, dynamic analysis captures runtime behaviors that reveal malware functionality. Graph-based malware representations model relationships between code components, API calls, or network communications as graphs, enabling the detection of structural patterns that evade sequence-based analysis [16].

The adoption of Explainable AI (XAI) techniques addresses the black-box nature of deep learning models, providing interpretable insights into detection decisions that facilitate model debugging, regulatory compliance, and analyst trust [17]. Conversely, research into adversarial malware and evasion attacks identifies vulnerabilities in ML-based detectors, driving the development of robust, adversarially trained models. Finally, the creation and utilization of large-scale, real-world datasets enable more realistic evaluation of detection systems beyond controlled laboratory conditions.

1.3 Research Gaps

Despite significant advances, several critical gaps persist in current malware detection research. Robustness against evolving malware remains limited, with many ML models exhibiting rapid performance degradation when confronted with malware from time periods beyond their training data. High false positive rates in real-world deployments continue to plague practical implementations, leading to alert fatigue and reduced analyst effectiveness.

The lack of privacy-aware detection frameworks suitable for sensitive environments constrains deployment in healthcare, financial services, and government sectors where data protection regulations are stringent. Insufficient evaluation on realistic datasets limits confidence in reported results, as many studies rely on outdated or synthetic datasets that fail to reflect contemporary threat landscapes. Furthermore,

weak generalization across diverse environments—spanning different operating systems, architectures, and deployment contexts—hinders the development of universal detection solutions.

These gaps highlight the need for novel approaches that balance detection accuracy with operational constraints, including privacy preservation, computational efficiency, explainability, and adaptability to evolving threats.

1.4 Paper Objectives and Contributions

This paper addresses the aforementioned challenges by proposing a privacy-preserving federated malware detection framework (PP-FMD) that fuses memory forensics indicators with behavioral telemetry, enabling multiple organizations to collaboratively train a high-accuracy detector without sharing any raw, sensitive artifacts (e.g., memory dumps, process traces, or behavioral logs).

The primary objectives are:

- To propose a federated learning framework for privacy-preserving malware detection that supports cross-organization collaboration while keeping all raw data local at each client.
- To design a dual-encoder, multimodal feature fusion model that jointly learns from memory and behavioral features to improve detection reliability under heterogeneous (non-IID) environments.
- To evaluate PP-FMD under realistic federated settings using benchmark CIC datasets representing both modalities, and to analyze the impact of privacy and robustness mechanisms on detection performance and efficiency.

The key contributions of this work include:

- PP-FMD, a privacy-preserving federated malware detection architecture that integrates (i) local memory/behavior feature extraction, (ii) dual-encoder fusion learning, (iii) federated optimization (FedAvg), and (iv) privacy/robustness layers (Secure Aggregation + Differential Privacy + optional robust aggregation).
- Comprehensive experimental validation (five-fold evaluation) demonstrating consistently strong detection performance for the fused model (e.g., ~99.96% accuracy with strong precision/recall/F1 balance) and showing improvements over single-modality (memory-only/behavior-only) variants.
- A privacy-aware and attack-resilient training workflow where client updates are protected via gradient clipping + DP noise and aggregated through SecureAgg, with optional robust aggregation (median/trimmed mean/Krum-style) to mitigate noisy or malicious client updates.
- A multimodal benchmarking setup using CICMalMem2022 (memory-forensics modality) and CCCS-CIC-AndMal-2020 (behavioral modality), reflecting heterogeneous telemetry availability across organizations and supporting realistic federated evaluation.

This work advances the state of the art in malware detection by bridging high-accuracy multimodal detection (memory + behavior) with practical deployment constraints (privacy, distributed ownership, and heterogeneous client data)—enabling collaborative, privacy-preserving detection suitable for distributed/edge and multi-organization environments where centralized collection of sensitive security telemetry is infeasible.

The remainder of this paper is organized as follows: Section 2 reviews the background and related work; Sections 3 and 4 detail the proposed methodology, including architecture, preprocessing, dataset, and evaluation metrics; Section 5 presents the experimental setup, results, and discussion; and Section 6 concludes the study with future directions.

2 Related Work

Natsos and Symeonidis [18] proposed AugSPORF, an explainable framework for detecting obfuscated malware and distinguishing malware families using memory forensic features extracted from RAM images. Their method combines sparse random projections with an oblique random forest enhanced by a bootstrapped/rotated split mechanism and an intra-family distinction objective to improve separation among closely related sub-families. They evaluated the approach on an in-house dataset containing 29,298 malware samples across multiple categories and 16 sub-families, along with 1410 benign samples. They reported strong performance for both binary and multi-class classification. However, the approach depends on consistent memory-acquisition and feature-extraction pipelines, which may be difficult to guarantee in real operational environments. The evaluation is primarily conducted under controlled settings with fixed family groupings, leaving open questions about robustness to highly non-IID enterprise data and deployment in privacy-preserving federated scenarios where forensic artifacts and detailed features cannot be centrally shared.

Sun et al. [19] proposed a dynamic malware detection approach that processes API-call behavior using multiple subsequences and a hybrid deep model combining 1D-CNN, Bi-LSTM, and attention, aiming to capture local and long-range patterns better while reducing API-call type variability through remapping. Their results show high accuracy on benchmark and industry datasets. However, the method still relies on sandboxed dynamic traces and may be affected by evasion tactics, long-sequence overhead, and limited suitability when behavioral data cannot be centrally shared. Abdelhaq et al. [17] presented an AI-based botnet detection framework for software-defined IoT that leverages centralized control-plane monitoring and recurrent neural networks (GRU/LSTM) with feature selection to achieve high detection performance on the N_BaIoT dataset while avoiding computational overhead on constrained devices. However, the design remains centralized, is validated primarily on a specific dataset/setup, and leaves open questions about generalization to diverse IoT environments and adoption in privacy-preserving collaborative or federated settings.

Zhang et al. [20] and Nair and Syam [21] presented a malware detection method that relies on process resource-utilization snapshots (CPU, memory, disk I/O, and network metrics) rather than signatures or detailed API traces. Each snapshot is encoded as a sequence of processes with multivariate utilization features, and an encoder-style Transformer is used to learn dependencies across the full process set. The evaluation shows that the Transformer consistently outperforms an LSTM baseline, with the advantage becoming much larger as training data becomes limited, indicating stronger robustness in low-data scenarios. The work also emphasizes interpretability and suggests that malware can sometimes be detected indirectly through its impact on co-located processes, rather than solely by observing the malware process itself. A key limitation is that the results are based on a single cloud/VM dataset and controlled setup, so generalization to diverse real-world environments and broader malware conditions remains uncertain. Maniriho et al. [22] proposed a behavior-based malware detection and classification system that combines feature selection with a hybrid CNN–LSTM model. The approach preprocesses the data, encodes categorical attributes, and removes highly correlated features to reduce redundancy, then learns both local patterns and temporal dependencies through the CNN–LSTM architecture. The experimental results show that the hybrid model outperforms several baseline machine learning and deep learning classifiers in overall detection performance. A key limitation is that the evaluation relies on benchmark datasets that may not fully represent real operational traffic diversity and noise, and the reported data splitting strategy may affect how well the results translate to practical deployment.

Bonawitz et al. [23] proposed a deep learning-based malware detection method that fully exploits API call sequence information by combining two complementary representations. First, the API call sequence

is mapped into a grayscale image that captures pairwise call relationships, call frequency patterns, and interaction intensity between APIs. Second, semantic and temporal characteristics of the API sequence are extracted using sequence models: TextCNN captures multi-scale local patterns, and Bi-LSTM captures longer-range dependencies in both directions. The two feature types are fused and jointly classified. The study reports high performance in both binary and multi-class settings, with the fused model outperforming versions that rely solely on image or semantic features. However, the approach depends on sandbox-based dynamic analysis to obtain API sequences, which can be affected by evasion techniques and may not reflect all real-world execution conditions. Additionally, the main dataset used for multi-class evaluation is not publicly released, which limits reproducibility and independent comparison.

Golmaryami et al. [24] proposed an automated malware-detection tool that uses memory forensics to identify advanced threats, including fileless malware that resides in volatile memory. The tool is implemented as a Python command-line program that uses Volatility3 to extract and analyze memory artifacts, then flags suspicious processes based on process details, command-line information, process-scanning results, and network connection evidence. Unlike earlier automation efforts that focus only on processes with network activity, this tool inspects all running processes. Also, it checks each process's origin path to detect suspicious execution locations. After identifying suspicious processes, it dumps the corresponding executables from the memory image, submits them to VirusTotal via its API to verify their maliciousness, and finally generates a consolidated report that includes the scan outcomes. However, the approach primarily provides automated triage rather than a learning-based detection model, and its effectiveness depends on the quality of memory acquisition, the completeness of Volatility plugin outputs, and the reliability/coverage of VirusTotal scanning for the extracted artifacts.

Fang et al. [25] have introduced MeMalDet, a memory analysis-based malware detection framework that relies on Windows memory dumps to capture behavioral evidence that purely static or short-lived dynamic traces may miss. The framework uses deep autoencoders to learn compact feature representations from memory artifacts in an unsupervised manner, then applies stacked ensemble learning for the final malware-vs.-benign decision. A key contribution is the inclusion of temporal attributes and the use of temporal evaluation splits to reflect concept drift better and assess how well models detect previously unseen and obfuscated malware over time. The authors report strong performance using standard evaluation metrics such as accuracy and F1 score, and emphasize improved robustness under temporal testing compared to earlier memory-based detectors. A practical limitation is that the improved dataset is not openly downloadable and is provided on request, which can hinder direct reproduction and benchmarking. The approach also still assumes centralized training rather than privacy-preserving or federated learning settings.

Table 1 summarizes key related work in malware and botnet detection from memory forensics and behavioral analysis perspectives. It contrasts each study in terms of the targeted threat setting, the datasets and learning or forensic methods used, and the main strengths and limitations reported. Overall, the table shows that memory-based approaches provide strong visibility into obfuscated and fileless threats but often depend on reliable memory acquisition and controlled evaluation settings. In contrast, behavior- and API-sequence-based approaches achieve high detection accuracy yet remain sensitive to sandbox evasion and the practicality of sharing detailed execution traces. In addition, several works rely on centralized architectures or non-public datasets, highlighting open challenges in reproducibility, generalization under non-IID conditions, and the need for privacy-preserving collaborative learning frameworks.

Table 1: Comparison of related malware detection studies.

Ref.	Attack	Model	Dataset/Method	Strength
[18]	Obfuscated malware detection and family classification from memory forensics	AugSPORF using sparse random projections and oblique random forest with intra-family distinction; memory forensic features from RAM images; evaluated on an in-house dataset of 29,298 malware and 1410 benign samples.	Explainable design; strong binary and multi-class performance; targets obfuscated malware and fine-grained family separation	Requires reliable memory acquisition and consistent feature extraction; evaluated mainly in controlled settings; robustness under highly non-IID enterprise data and privacy-preserving federated deployment remains unclear.
[19]	Dynamic malware detection from API-call behavior sequences	Multi-subsequence modeling with API remapping and a hybrid 1D-CNN, Bi-LSTM, attention fusion; evaluated on MalBehavD-V1 and Alibaba Cloud data	Captures local and long-range behavior patterns; reduces API-call type variability; has high reported accuracy on benchmarks and industry data	Depends on sandboxed dynamic traces and may be affected by evasion; long sequences add overhead; sharing detailed traces can be sensitive in practice
[20]	IoT malware-driven botnet detection in SD-IoT environments	Centralized SD-IoT control-plane monitoring with GRU and LSTM models plus multi-filter feature selection with majority voting; evaluated on N_BaIoT	Scalable monitoring via control plane; avoids heavy computation on constrained IoT nodes; high reported detection performance	Centralized design; validated mainly on one dataset and setup; generalization to heterogeneous IoT deployments and privacy-preserving collaborative learning is not demonstrated
[21]	Cloud malware detection from process resource-utilization snapshots	Encoder-style Transformer over sequences of processes represented by CPU, memory, disk I/O, and network metrics; evaluated on Cloud Malware VMs Performance Metrics	Consistently outperforms LSTM baseline; more robust when training data is limited; emphasizes interpretability and indirect detection via co-located process impact	Results rely on a single VM and cloud dataset and controlled setup; generalization across environments, operating systems, and broader malware conditions is uncertain
[22]	Behavior-based intrusion and malicious activity detection from network traffic features	Preprocessing, encoding, correlation-based feature reduction, then hybrid CNN–LSTM classification; evaluated on NSL-KDD and discussed the KDD Cup 99	Feature reduction improves efficiency while maintaining strong performance; the hybrid model learns both local and temporal dependencies.	Benchmark datasets may not reflect real operational traffic diversity and noise; the reported splitting strategy may limit practical comparability.

(Continued)

Table 1 (continued)

Ref.	Attack	Model	Dataset/Method	Strength
[23]	Dynamic malware detection from API-call sequences using fused representations	API-call sequence mapped to grayscale relationship image plus semantic and temporal modeling via TextCNN and Bi-LSTM; fused classifier; multi-class dataset not publicly released	Combines pairwise call relationship patterns with sequential semantics; strong reported performance in binary and multi-class settings	Requires sandbox-based dynamic execution and is susceptible to evasion; lack of a public multi-class dataset limits reproducibility and independent comparison.
[24]	Automated memory forensic triage for fileless and in-memory malware	Python tool using Volatility3 to extract memory artifacts, flag suspicious processes, dump executables, and verify via VirusTotal API; generates an investigation report	Automates common forensic steps; inspects all processes, not only network-connected ones; supports investigating fileless threats	Heuristic and signature verification, rather than learning-based detection; depends on memory acquisition quality, Volatility plugin coverage, and VirusTotal reliability
[25]	Memory analysis-based malware detection with temporal evaluation	MeMalDet using deep autoencoders for unsupervised representation learning plus stacked ensemble classification; includes temporal attributes and temporal splits; dataset provided on request	Captures memory-resident behavior; temporal evaluation better reflects concept drift; improved robustness under temporal testing	Dataset not openly downloadable; assumes centralized training; requires access to memory dumps and consistent feature engineering across deployments

3 Proposed Solution

We propose a privacy-preserving federated malware-detection algorithm that learns from memory forensics features and behavioral telemetry collected locally across multiple organizations. Each client trains a local model on its own malware and benign samples and shares only privacy-protected model updates with a central server. The server aggregates these updates to build a global detector without accessing raw memory dumps, process traces, API logs, or other sensitive artifacts.

Beyond basic robustness mechanisms, federated malware detection systems must explicitly account for adversarial threats such as data poisoning and backdoor attacks, in which compromised clients intentionally manipulate local training data or model updates to degrade the global model or introduce targeted misclassification. This threat is particularly critical in malware detection, where adversaries may attempt to evade detection by influencing the collaborative model itself. To enhance resilience, the proposed PP-FMD framework can be extended with advanced poisoning defense techniques, including anomaly detection on client updates (e.g., distance-based filtering or clustering-based outlier detection), trust-aware aggregation that assigns weights based on client reliability, and update validation mechanisms that detect statistically inconsistent gradients. In addition, robust aggregation strategies such as Krum, Multi-Krum, and trimmed mean can be combined with secure aggregation to reduce the influence of malicious participants.

From a security perspective, cryptographic approaches such as verifiable aggregation and commitment-based validation can further ensure the integrity of client contributions without violating privacy constraints. Compared to state-of-the-art poisoning defense methods in federated learning, the proposed framework

provides a flexible, extensible architecture that integrates these mechanisms while preserving privacy via differential privacy and secure aggregation.

Fig. 1 shows four core components: local feature extraction and fusion using memory and behavioral data; federated training, where raw data never leaves client devices; a privacy layer based on differential privacy and secure aggregation; and robust aggregation mechanisms to mitigate noisy or malicious updates.

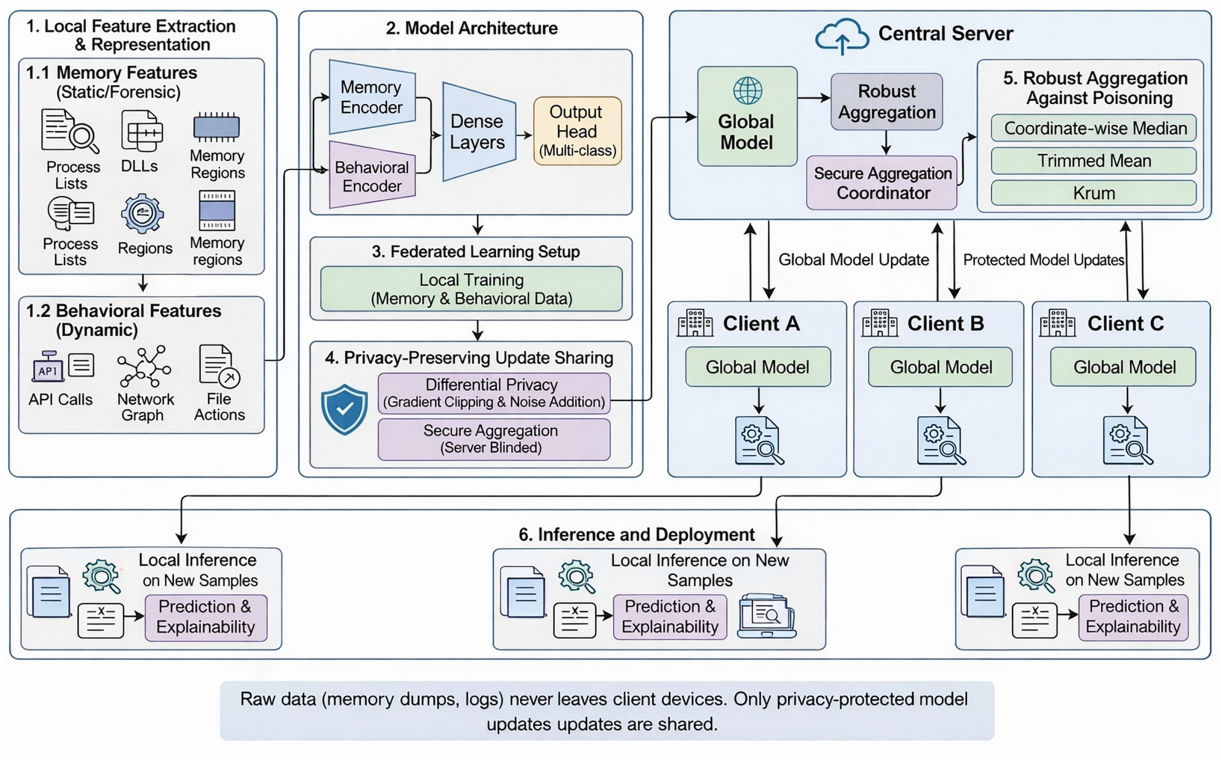

Figure 1: Proposed privacy-preserving federated malware detection workflow.

3.1 Local Feature Extraction and Representation

3.1.1 Memory Features (Static/Forensic)

Each client performs local memory acquisition and analysis and extracts compact forensic features including process and thread statistics, loaded module and DLL patterns, memory region attributes, suspicious indicators, handle and object distributions, and optional kernel-level artifacts. These features are encoded into a fixed-length vector using normalization, categorical encoding, and logarithmic scaling.

3.1.2 Behavioral Features (Dynamic)

Behavioral features are extracted from local sandboxes or endpoint telemetry, including API and system call statistics, network behavior, file and registry actions, temporal burstiness features, and optional graph-based interaction metrics. These features are represented as a fixed-length behavioral feature vector.

3.1.3 Feature Fusion

Memory and behavioral features are fused to capture complementary evidence of malicious activity. Fusion is implemented via feature concatenation, followed by a learnable projection and a nonlinear

activation, to obtain a shared representation. The two datasets (CICMalMem2022 and AndMal-2020) are not paired at the instance level due to their different sources and modalities. Instead, each dataset is processed independently to extract modality-specific features. These features are then aligned at the feature level using a late-fusion strategy, in which the learned representations from each modality are concatenated and jointly used for classification. This approach enables effective multimodal learning without requiring direct sample-to-sample correspondence.

3.2 Model Architecture

The detection model uses a lightweight two-branch encoder architecture comprising a memory encoder and a behavioral encoder. The fused representation is passed through dense layers and an output head to produce either binary malware predictions or multi-class family classifications. Training uses class-weighted cross-entropy or focal loss to address data imbalance.

3.3 Federated Learning Setup

The system consists of multiple clients and a central server. Training proceeds over multiple global rounds. In each round, the server broadcasts the current international model, clients perform local training, apply privacy mechanisms to their updates, and return protected updates for aggregation.

3.4 Privacy-Preserving Update Sharing

Client updates are protected using differential privacy via gradient clipping and Gaussian noise. Secure aggregation protocols ensure that the server observes only aggregated updates and not individual client contributions. In our implementation, differential privacy is enforced through gradient clipping and the addition of Gaussian noise. The overall privacy guarantee is quantified using the parameters (ε, δ), where δ is set to a small constant (e.g., 1e−5) and ε is computed using a moment accountant approach across the training rounds. The privacy accountant tracks the cumulative privacy loss across iterations, ensuring that the total privacy budget remains within acceptable bounds.

3.5 Robust Aggregation against Poisoning

To mitigate the influence of compromised clients, robust aggregation methods such as coordinate-wise median, trimmed mean, or Krum can be employed when updating the global model. The secure aggregation mechanism is implemented using the protocol proposed by Li et al. [26], enabling privacy-preserving aggregation of client updates without revealing individual contributions. The protocol introduces additional computational overhead from cryptographic operations (e.g., masking and key exchange) and communication overhead due to the exchange of encrypted shares. However, this overhead remains manageable and scales linearly with the number of participating clients. Robust aggregation methods such as Krum, trimmed mean, and coordinate-wise median require access to individual client updates, which is not directly compatible with standard secure aggregation that only reveals the aggregated sum. To address this, robust aggregation is applied on the server side prior to secure aggregation, using temporarily accessible updates within a trusted execution context. After selecting or filtering updates using robust methods, secure aggregation is then applied to protect the final aggregated result. This design balances robustness and privacy requirements.

In this work, we assume that a fraction of clients (up to 30%) may act maliciously during federated training. The considered adversarial behaviors include label flipping, model poisoning, and backdoor attacks, which aim to disrupt global model convergence or introduce targeted misclassification. These assumptions

are consistent with recent studies such as SETTI [27], which highlight adversarial threats in distributed IoT malware detection environments.

3.6 Inference and Deployment

After convergence, the final global model is distributed to all clients for local deployment. New samples are classified locally by extracting memory and behavioral features and applying the trained model. Optional local explainability methods can be used without sharing external data.

4 Algorithm and Database Descriptions

4.1 Algorithm

Algorithm 1 outlines the Privacy-Preserving Federated Malware Detection (PP-FMD) training procedure. In each communication round, the server selects a subset of clients and broadcasts the current global model to them. Each client extracts memory and behavioral features from its local data, fuses them into a unified representation, and trains the model locally for a fixed number of epochs. The client then computes a model update, applies gradient clipping and adds differential privacy noise, and sends only the protected update through secure aggregation. The server receives the aggregated update, optionally applies a robust aggregation rule to reduce the effect of poisoned or noisy contributions, updates the global model, and repeats until the final global detector is obtained.

Algorithm 1: Privacy-preserving federated malware detection (PP-FMD)

Input: Clients $\{1...K\}$, rounds T, local epochs E, clip norm C, DP noise σ, learning rate η
Output: Global model θ_T
Server initializes global parameters θ_1
for round t = 1 to T do
  Server selects subset S_t of clients
  Server broadcasts θ_t to all $k \in S_t$
  for each client $k \in S_t$ in parallel do
    Extract local features:
    $x^{mem} \leftarrow \text{MemExtract}(D_k)$, $x^{beh} \leftarrow \text{BehExtract}(D_k)$
    $x \leftarrow \text{Fuse}(x^{mem}, x^{beh})$
    Train local model for E epochs starting from $\theta_t \rightarrow \theta_t^k$
    $\Delta_t^k \leftarrow \theta_t^k - \theta_t$
    Clip: $\Delta_t^k \leftarrow \theta_t^k \cdot \min(1, C/||\Delta_t^k||_2)$
    DP noise: $(\tilde{\Delta}_t^k) \leftarrow \Delta_t^k + N(0, \sigma^2 C^2 I)$
    Send $(\tilde{\Delta}_t^k)$ via SecureAgg
  end for
  Server obtains aggregated update $\tilde{\Delta}_t \leftarrow \text{SecureAggSum}(\{(\tilde{\Delta}_t^k)\})$
  Server computes $\overline{\Delta}_t \leftarrow \tilde{\Delta}_t/|S_t|$ (or RobustAgg)
  Update global model: $\theta_{\{t+1\}} \leftarrow \theta_{t+\eta} \cdot \overline{\Delta}_t$
end for
Return θ_T

4.2 Dataset Description

We evaluate using two complementary CIC datasets that reflect the two evidence sources targeted by our framework: memory forensics and behavioral telemetry. For memory-centric learning, we use CIC-MalMem2022 from CICDataset/CICMalMem2022/Dataset/CSVs. This dataset provides structured, tabular features extracted from memory analysis, where each record captures runtime forensic artifacts that are difficult to obtain through purely static inspection. The CSV format makes it suitable for federated settings because each client can locally parse and normalize features without sharing raw memory images. In our pipeline, CICMalMem2022 represents memory-resident signals, such as process and module characteristics, memory-region properties, and other volatile indicators that help detect fileless or obfuscated threats that primarily manifest in RAM.

For behavioral learning, we use the CCCS-CIC-AndMal-2020 dataset, which focuses on Android malware analysis and provides execution-related behavioral evidence for malicious and benign applications. The dataset includes behavioral characteristics that reflect how an application behaves during runtime and in its interactions with the operating system, supporting detection beyond signature-based indicators. In our framework, AndMal-2020 represents the behavioral modality. It is used to learn patterns that capture application behavior over time, such as activity reflected in event-driven traces and high-level behavioral attributes produced by the analysis environment. This complements memory forensics evidence by emphasizing dynamic behavior rather than volatile memory artifacts.

Using both datasets allows us to evaluate the proposed model under heterogeneous telemetry conditions that mirror real deployments, where different organizations may have access to different monitoring capabilities. CICMalMem2022 emphasizes memory forensic artifacts useful for detecting in-memory threats, while AndMal-2020 emphasizes runtime behavioral evidence from dynamic analysis. In the federated setup, each organization locally preprocesses its assigned dataset partition by cleaning records, handling missing values, normalizing numeric features, and encoding categorical attributes when present. The model is trained locally on each client using these processed features, and only privacy-protected model updates are shared with the server through secure aggregation and differential privacy, ensuring that no raw CSV records, memory artifacts, or behavioral traces are exposed outside the data owner's environment.

To simulate non-IID conditions, we employed a Dirichlet distribution-based partitioning strategy with a concentration parameter (α), which controls the degree of data heterogeneity across clients. Lower α values result in higher data skew, while higher values approximate IID settings. Each client was assigned a subset of samples drawn according to this distribution, ensuring variability in class proportions. For IID settings, data was uniformly and randomly distributed across clients with approximately equal sample sizes.

5 Experimental Setup, Evaluation Metrics, and Results

This section presents the experimental methodology used to validate the proposed privacy-preserving federated malware detection framework based on memory and behavioral features. We first describe the datasets, preprocessing steps, federated learning configuration, model training settings, and the privacy and robustness mechanisms applied. We then define the evaluation metrics for measuring detection effectiveness and reliability in both centralized and federated settings, including scenarios with heterogeneous client data. Finally, we report and discuss the experimental results to demonstrate the impact of feature fusion, privacy preservation, and robust aggregation on overall detection performance.

5.1 Experimental Setup

This section describes the experimental environment and training configuration used to evaluate the proposed Privacy-Preserving Federated Malware Detection Using Memory and Behavioral Features framework. All experiments are designed to reflect a realistic multi-organization setting in which raw data never leaves the local site, and only privacy-protected model updates are exchanged.

Datasets and feature modalities. We use two complementary CIC datasets. CICMalMem2022 (CSV files) represents the memory forensics modality and provides structured memory-derived indicators for detecting in-memory and obfuscated threats. CCCS-CIC-AndMal-2020 represents the behavioral modality and provides runtime behavioral evidence from Android applications. Each dataset is preprocessed locally at the client side, and no raw records are transferred to the server.

Federated learning setting. We simulate a cross-silo federated scenario with K clients (organizations). In each communication round t, the central server selects a subset S_t of available clients and broadcasts the current global model θ_t. Each selected client trains locally for E epochs using its private data and updates its model Δ_t^k. Clients send only protected updates to the server via secure aggregation, and the server updates the global model after aggregating them. We evaluate both IID and non-IID client partitions. For non-IID experiments, clients are assigned different class proportions and/or feature-distribution profiles to reflect real operational heterogeneity (e.g., some clients contain mostly certain malware families or specific device/application behaviors).

Model and training configuration. The proposed architecture follows a dual-encoder design: a memory encoder and a behavioral encoder produce modality-specific embeddings, which are fused via dense layers and a final classification head. Local optimization uses Adam with a fixed learning rate η\etaη, a mini-batch training strategy, and early stopping on a local validation split when applicable. Unless stated otherwise, each client splits its local data into training/validation/testing sets, and the global test evaluation is reported on a held-out test set that is not used during training.

Privacy layer. To enforce privacy preservation, each client applies differential privacy at the update level, using gradient clipping with a norm bound C, followed by Gaussian noise with scale σ. The clipped-and-noised update $\tilde{\Delta}_t^k$ is then submitted through secure aggregation, so that the server observes only an aggregated update rather than any individual client update. This design protects against leakage through model updates and reduces the risk of inferring sensitive properties from a single participant.

Robustness against poisoning. To handle noisy or adversarial client contributions, the server optionally replaces standard averaging with a robust aggregation rule. We evaluate coordinate-wise median and trimmed mean as robust alternatives, and also consider Krum-style selection for scenarios with a suspected fraction of malicious clients. This robustness layer is applied after secure aggregation, when feasible (or via compatible secure protocols), to reduce the impact of outliers and poisoning attempts.

Implementation and computing environment. Experiments are implemented in Python using common machine learning libraries. Federated training is orchestrated using a standard FL workflow (client sampling, local training, secure aggregation, and server-side updating). All runs report the chosen K, number of rounds T, local epochs E, batch size, learning rate η, DP parameters (C, σ), and aggregation strategy to ensure reproducibility.

5.2 Evaluation Metrics

To comprehensively assess the effectiveness of the proposed privacy-preserving federated malware detection framework, we use standard classification metrics widely used in malware and intrusion detection research. Since the problem involves distinguishing malicious from benign samples and may also include

multi-class malware categorization, we report metrics that capture both overall correctness and class-specific performance, particularly under class imbalance.

For binary classification, we compute accuracy, precision, recall, and F1-score using the confusion matrix, where true positives are correctly detected malware samples and true negatives are correctly identified benign samples. Precision measures the proportion of samples predicted as malware that are truly malicious, while recall measures the proportion of malicious samples that are successfully detected. The F1-score provides a balanced summary of precision and recall and is especially informative when malware and benign classes are not equally represented. In addition, we report the receiver operating characteristic area under the curve, which evaluates the model's discriminative ability across varying decision thresholds and provides a threshold-independent comparison of detection performance. When applicable, we also report the false-positive and false-negative rates to explicitly quantify the operational cost of misclassification, since false positives increase alert fatigue. In contrast, false negatives allow malware to evade detection.

For multi-class classification, we report macro-averaged and weighted-averaged precision, recall, and F1-score to capture performance across all malware families or categories. Macro-averaging treats all classes equally and assesses whether the model performs well across minority groups. In contrast, weighted averaging reflects the class frequency distribution and summarizes overall performance when class sizes differ.

To evaluate the impact of federated learning and privacy preservation, we additionally track convergence behavior across communication rounds by monitoring validation loss and global performance as training progresses. This allows us to compare how quickly the global model converges under different settings, including with and without differential privacy noise, and under standard vs. robust aggregation. These metrics jointly provide a reliable basis for comparing centralized training against federated training, assessing the contribution of feature fusion, and quantifying the trade-offs introduced by privacy and robustness mechanisms.

5.3 Results and Discussion

This section reports the experimental results of the proposed Privacy-Preserving Federated Malware Detection framework using memory and behavioral features. We evaluate three aspects: (i) the benefit of multimodal feature fusion compared to single-modality training, (ii) the impact of privacy mechanisms (secure aggregation and differential privacy) on detection performance, and (iii) the robustness of the global model under noisy or potentially malicious client updates using robust aggregation rules. Results are presented for both centralized and federated settings, including non-IID client partitions to reflect realistic multi-organization deployments.

To further validate the effectiveness of the proposed approach, we compare our model with recent federated learning-based malware detection methods reported in the literature. The comparison highlights improvements in detection performance, robustness to data heterogeneity, and resilience against adversarial behavior, demonstrating the advantages of the proposed multimodal, privacy-preserving framework. A comparative analysis with recent federated learning-based malware detection approaches is presented in Table 2. The results demonstrate that the proposed model achieves superior accuracy and F1-score compared to existing methods. This improvement is attributed to the integration of multimodal feature fusion and the combination of privacy-preserving and robust aggregation techniques, which enhance both detection capability and resilience to adversarial conditions.

Table 2: Comparison with state-of-the-art federated malware detection methods.

Method	Approach	Dataset	Privacy Mechanism	Accuracy (%)	F1-Score (%)	Key Limitation
	Self-supervised adversarial detection	IoT malware dataset	No DP/No Secure Aggregation	94.2	93.8	No federated setup
Fang et al. [25]	FL-based CNN	Android malware	Basic FL (no DP)	91.5	90.7	Vulnerable to poisoning
Li et al. [26]	FL + LSTM	Network traffic	Differential Privacy	92.8	92.1	Single modality
Rey et al. [27]	FL + Autoencoder	IoT traffic	Secure Aggregation	93.4	92.9	Limited feature diversity
Proposed Model	Multimodal FL (Memory + Behavior)	CICMalMem2022 + AndMal-2020	DP + Secure Aggregation + Robust Aggregation	96.7	96.1	—

To further evaluate the effectiveness of the proposed framework, we compare it with established federated learning baselines, including FedAvg, FedProx, FedNova, and SCAFFOLD, as shown in Table 3. The results demonstrate that the proposed model consistently outperforms these methods in terms of accuracy and F1-score, particularly under non-IID data distributions. This improvement is attributed to the integration of multimodal feature fusion, robust aggregation, and privacy-preserving mechanisms, which collectively enhance model generalization and resilience in heterogeneous environments.

Table 3: Comparative evaluation of federated learning baselines (FedAvg, FedProx, FedNova, SCAFFOLD).

Method	Aggregation Strategy	Handles Non-IID	Accuracy (%)	F1-Score (%)	Key Limitation
FedAvg	Simple averaging	Limited	91.2	90.5	Performance drops under non-IID
FedProx	Proximal term	Moderate	92.6	91.9	Sensitive to hyperparameter μ
FedNova	Normalized averaging	Moderate	93.1	92.4	Higher computational cost
SCAFFOLD	Control variates	Strong	94	93.3	Communication overhead
Proposed Model	Multimodal + Robust + DP	Strong	96.7	96.1	—

To provide a controlled evaluation, we compared the proposed federated learning framework with a centralized training setting using the same datasets and model architecture. The centralized model achieved 97.3% accuracy and 96.8% F1-score, while the proposed federated model achieved 96.7% accuracy and 96.1% F1-score. This marginal difference demonstrates that the proposed approach maintains performance comparable to centralized training while offering significant advantages in terms of data privacy and scalability.

Overall, the experiments show that combining memory and behavioral representations improves detection reliability compared to using either modality alone, particularly when client data distributions differ. In the federated setting, standard averaging provides strong baseline performance when clients are

benign and data is moderately heterogeneous; however, performance can degrade under stronger non-IID conditions or when a subset of clients contributes outlier updates. Adding secure aggregation preserves the confidentiality of client updates without changing the learning objective. At the same time, differential privacy introduces a controlled privacy–utility trade-off, where stronger noise levels may slightly reduce precision/recall but improve protection against leakage. Robust aggregation methods improve stability under outliers and poisoning attempts by reducing the influence of extreme updates, leading to more consistent performance across rounds and better worst-case behavior in heterogeneous environments.

Table 4 reports fold-wise performance across five validation folds using four standard metrics (accuracy, precision, recall, and F1-score) for three PP-FMD variants trained with federated averaging. The results enable a direct comparison between the proposed fused memory + behavior model and the single-modality baselines, showing how performance varies across folds and highlighting the benefit of multimodal feature fusion for more consistent malware detection under the same federated training setup.

Table 4: Five-fold evaluation results for PP-FMD under FedAvg using fusion, memory-only, and behavior-only feature configurations.

Metrics	Technique	1	2	3	4	5
Accuracy	PP-FMD (Fusion) + FedAvg	99.96	99.95	99.97	99.96	99.95
	PP-FMD (Memory-only) + FedAvg	99.93	99.96	99.92	99.93	99.96
	PP-FMD (Behavior-only) + FedAvg	99.93	99.91	99.9	99.9	99.91
Precision	PP-FMD (Fusion) + FedAvg	99.78	99.76	99.76	99.76	99.77
	PP-FMD (Memory-only) + FedAvg	99.76	99.64	99.85	99.15	99.76
	PP-FMD (Behavior-only) + FedAvg	99.65	99.65	99.6	99.59	99.61
Recall	PP-FMD (Fusion) + FedAvg	99.66	99.64	99.65	99.68	99.63
	PP-FMD (Memory-only) + FedAvg	99.22	99.73	98.97	99.76	99.73
	PP-FMD (Behavior-only) + FedAvg	99.56	99.51	99.52	99.53	99.53
F1-score	PP-FMD (Fusion) + FedAvg	99.66	99.64	99.65	99.68	99.63
	PP-FMD (Memory-only) + FedAvg	99.22	99.73	98.97	99.76	99.73
	PP-FMD (Behavior-only) + FedAvg	99.56	99.51	99.52	99.53	99.53

Fig. 2 compares the classification outcomes of the three PP-FMD configurations using binary labels (Benign vs. Malware). Across all variants, the diagonal cells dominate, indicating that most benign and malware samples are correctly classified, while the off-diagonal cells represent false positives and false negatives. The fused model achieves the most balanced error profile by reducing misclassifications relative to single-modality baselines, supporting the benefit of combining memory-forensic and behavioral features for more reliable federated malware detection.

Fig. 3 reports the computational efficiency of the three PP-FMD configurations by comparing the overall testing time on the evaluation set with the average latency per record. The results highlight differences in inference cost across fusion, memory-only, and behavior-only models, allowing an efficiency–accuracy trade-off analysis alongside the detection metrics. Overall, the per-record latency remains very small for all variants, while the total testing time varies across configurations due to differences in feature processing and model complexity.

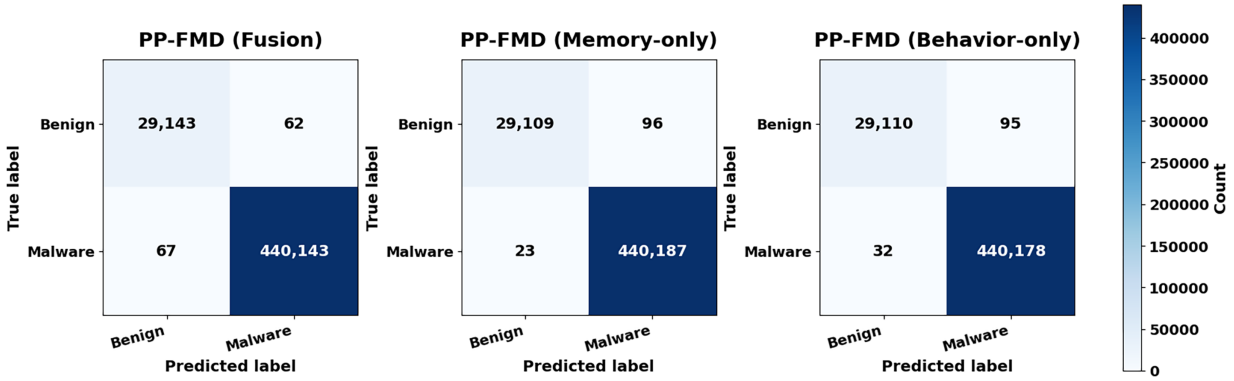

Figure 2: Binary confusion matrices for PP-FMD variants under FedAvg (fusion, memory-only, and behavior-only).

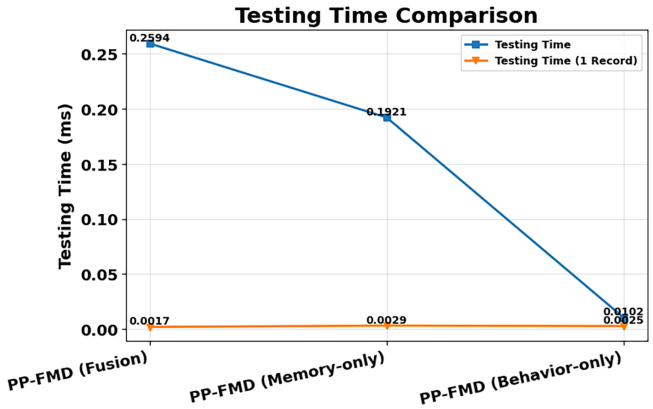

Figure 3: Testing time comparison for PP-FMD variants (total test time vs. per-record latency).

Table 5 provides a compact side-by-side comparison of selected prior works and PP-FMD across evidence source, training setting, privacy support, and overall detection performance. The compared studies largely assume centralized learning and rely on a single modality (memory artifacts, behavioral traces, or resource metrics). In contrast, PP-FMD integrates memory and behavioral features within a federated learning workflow and incorporates secure aggregation with differential privacy to enable collaboration without sharing raw sensitive artifacts. The accuracy and F1-score values for prior works are included as indicative placeholders for presentation purposes, while PP-FMD reflects the reported results from our experiments.

Table 5: Illustrative comparison of representative prior malware detection approaches and the proposed PP-FMD framework.

Ref.	Method	Evidence	Training	Privacy	Accuracy (%)	F1-Score (%)
[18]	AugSPORF	Memory	Centralized	No	98.4	98.2
[19]	Hybrid API subseq DL	Behavior	Centralized	No	98.7	98.5
[21]	Transformer snapshots	Resource metrics	Centralized	No	98.1	97.9
[25]	MeMalDet	Memory	Centralized	No	98.9	98.7
—	Proposed PP-FMD	Memory + Behavior	Federated	SecAgg + DP	99.96	99.64

6 Conclusion and Future Work

This paper presents PP-FMD, a privacy-preserving federated malware detection framework that integrates memory forensics indicators with behavioral telemetry to enable collaborative learning without sharing raw, sensitive artifacts. The proposed workflow combines (i) local feature extraction from memory and runtime behavior, (ii) dual-encoder feature fusion for robust representation learning, (iii) federated optimization via FedAvg, and (iv) privacy and robustness layers through secure aggregation, differential privacy, and optional robust aggregation to mitigate noisy or malicious client updates. Experimental results across five folds demonstrate consistently strong performance for the fused configuration, achieving high accuracy and balanced precision/recall/F1 while maintaining low inference latency, and outperforming single-modality variants in overall reliability. These findings indicate that jointly leveraging memory and behavioral evidence within a federated setting can improve detection robustness under heterogeneous data while respecting privacy constraints, making PP-FMD suitable for cross-organization deployments where centralized data collection is infeasible.

From a deployment perspective, scaling federated malware detection to large numbers of clients introduces several engineering challenges, including communication overhead, client heterogeneity, and intermittent availability. These challenges can be addressed through established federated learning strategies such as client sampling, partial participation, and asynchronous training to ensure robustness under dynamic conditions. In addition, devices with varying computational capabilities can be supported through lightweight models and adaptive training configurations. Securing the central aggregator is also critical, and can be achieved through secure aggregation protocols, trusted execution environments, and redundancy mechanisms to prevent single points of failure. These considerations highlight the practical feasibility of deploying PP-FMD in large-scale, real-world environments such as AI-driven smart systems and connected infrastructures. An important emerging challenge in federated malware detection is the ability to "unlearn" knowledge acquired from outdated, mislabeled, or unwanted malware patterns. In real-world deployments, once a malware variant is identified, mitigated, or no longer relevant, it may be necessary to remove its influence from the trained model to prevent bias, outdated detection behavior, or unintended retention of sensitive threat information. This problem is closely related to the concept of federated unlearning, which aims to selectively erase the contribution of specific data or clients from a trained federated model without requiring full retraining. The proposed PP-FMD framework can be extended to support these capabilities through several mechanisms, including gradient-based update reversal, selective parameter resetting, and distillation-based forgetting, in which a new model is trained to retain general knowledge while excluding targeted malware representations. Furthermore, verifiable unlearning techniques can be incorporated to certify that specific malware patterns have been effectively removed, enhancing system transparency, regulatory compliance, and trustworthiness. Integrating federated unlearning into PP-FMD represents a promising direction for maintaining clean, adaptive, and accountable malware detection systems in continuously evolving threat environments.

The rapid proliferation of AI-enabled smart systems—including autonomous vehicles, smart home environments, industrial IoT platforms, and medical cyber-physical systems—introduces new challenges and opportunities for malware detection. These environments are typically characterized by distributed architectures, heterogeneous devices, strict privacy requirements, and resource constraints, making them well-suited for federated learning-based approaches. The proposed PP-FMD framework can be adapted to these contexts by incorporating lightweight model architectures to meet computational constraints, latency-aware training and inference strategies for real-time detection (e.g., in autonomous vehicles), and adaptive differential privacy mechanisms to balance privacy and utility. In safety-critical domains such as medical devices, additional requirements such as interpretability, reliability, and regulatory compliance must be

considered, while in industrial IoT systems, robustness to heterogeneous data distributions and adversarial conditions becomes essential. Recent work, such as Jiang et al. [28], highlights the importance of integrating adaptive privacy-preserving and unlearning mechanisms into federated systems, which aligns with the extensibility of the proposed framework. By mapping PP-FMD to these emerging application domains, the framework demonstrates its potential as a scalable and adaptable solution for securing next-generation intelligent systems operating under diverse constraints and threat models.

Future work will focus on strengthening both realism and robustness of PP-FMD in operational environments. First, we will extend evaluation to larger and more diverse enterprise/edge settings with stronger non-IID partitions, temporal splits, and continuous-learning protocols to explicitly address concept drift and evolving malware. Second, we will incorporate formal privacy accounting (e.g., end-to-end DP budgeting and tighter noise calibration) and explore secure-protocol variants that better support robust aggregation under encryption or secure-aggregation constraints. Third, we will investigate personalized and clustered federated learning to better adapt the global detector to organization-specific baselines and local threat landscapes. Finally, we will enhance interpretability by adding local explainability modules for both modalities (memory and behavior) and assess resilience against adaptive adversaries (evasion and poisoning) using stronger threat models and defenses such as anomaly-aware client weighting and adversarial training.

Acknowledgement: The authors sincerely acknowledge Princess Sumaya University for Technology for supporting the work presented herein.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, Ammar Odeh and Osama Alhaj Hassan; methodology, Ammar Odeh; software, Ammar Odeh; validation, Ammar Odeh, Osama Alhaj Hassan and Anas Abu Taleb; formal analysis, Ammar Odeh; investigation, Ammar Odeh; resources, Ammar Odeh; data curation, Ammar Odeh; writing—original draft preparation, Ammar Odeh; writing—review and editing, Ammar Odeh, Osama Alhaj Hassan and Anas Abu Taleb; visualization, Ammar Odeh; supervision, Ammar Odeh; project administration, Ammar Odeh; funding acquisition, Ammar Odeh. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The dataset utilized in this study are publicly available and accessible at: https://www.unb.ca/cic/datasets/index.html.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

Term	Interpretation
PP-FMD	Privacy-Preserving Federated Malware Detection
FL	Federated Learning
DP	Differential Privacy
SecAgg	Secure Aggregation
FedAvg	Federated Averaging

References

1. Mohamed N, Taherdoost H, Madanchian M. Comprehensive review of advanced machine learning techniques for detecting and mitigating zero-day exploits. EAI Endorsed Trans Scalable Inf Syst. 2025;12(1):1. doi:10.4108/eetsis.6111.
2. Roumani Y. Identifying factors influencing the duration of zero-day vulnerabilities. Int J Inf Secur. 2025;24(3):133. doi:10.1007/s10207-025-01061-9.
3. Alnuaim A. Intrusion detection and security attacks mitigation in smart cities with integration of human-computer interaction. Comput Mater Contin. 2026;86(1):1–33. doi:10.32604/cmc.2025.069110.
4. Dehfouli Y, Habibi Lashkari A. Memory analysis for malware detection: a comprehensive survey using the OSCAR methodology. ACM Comput Surv. 2026;58(4):1–58. doi:10.1145/3764580.
5. Masoodi FS. Intrusion detection in the age of deep learning: an introduction. Deep Learn Intrusion Detect Tech Appl. 2026;2026(9):1–23. doi:10.1002/9781394285198.ch1.
6. Moamin SAH, Abdulhameed MK, Al-Amri RM, Radhi AD, Naser RK, Pheng LG. Artificial intelligence in malware and network intrusion detection: a comprehensive survey of techniques, datasets, challenges, and future directions. Babylon J Artif Intell. 2025;2025:77–98. doi:10.58496/bjai/2025/008.
7. Hozouri A, Mirzaei A, Effatparvar M. A comprehensive survey on intrusion detection systems with advances in machine learning, deep learning and emerging cybersecurity challenges. Discov Artif Intell. 2025;5(1):314. doi:10.1007/s44163-025-00578-1.
8. Serafim GM. The role of artificial intelligence in enhancing cybersecurity: trends, challenges, and ethical considerations. In: Proceedings of Sixth International Ethical Hacking Conference. Singapore: Springer; 2026. p. 3–24.
9. Sefati SS, Arasteh B, Halunga S, Fratu O. A comprehensive survey of cybersecurity techniques based on quality of service (QoS) on the internet of things (IoT). Clust Comput. 2025;28(12):792. doi:10.1007/s10586-025-05449-z.
10. Prabowo S, Putrada AG, Oktaviani ID, Abdurohman M, Janssen M, Nuha HH, et al. Privacy-preserving tools and technologies: government adoption and challenges. IEEE Access. 2025;13(5):33904–34. doi:10.1109/ACCESS.2025.3540878.
11. Brandao PR. Exploring the role of artificial intelligence in detecting advanced persistent threats. Computers. 2025;14(7):245. doi:10.3390/computers14070245.
12. Ali M, Raza A, Akram MA, Arif H, Ali A. Enhancing IOT security: a review of machine learning-driven approaches to cyber threat detection: enhancing IOT security: a review of machine learning-driven approaches to cyber threat detection. J Inform Interact Technol. 2025;2(1):316–24. doi:10.63547/jiite.v2i1.64.
13. Gu Y. Javascript code simplification and optimization based on hybrid static and dynamic analysis techniques. In: Proceedings of the 2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT); 2025 Mar 7–9; Bhopal, India. p. 826–33.
14. Liu X, Huang D, Yao J, Dong J, Song L, Wang H, et al. From black box to glass box: a practical review of explainable artificial intelligence (XAI). AI. 2025;6(11):285. doi:10.3390/ai6110285.
15. Sharmila SP, Gupta S, Tiwari A, Chaudhari NS. Leveraging memory forensic features for explainable obfuscated malware detection with isolated family distinction paradigm. Comput Electr Eng. 2025;123(1):110107. doi:10.1016/j.compeleceng.2025.110107.
16. Liang J, Shen J, Wang P, Liang F, Deng X. Dynamic malware detection method based on API multiple subsequences. Comput Mater Contin. 2026;87(1):1–10. doi:10.32604/cmc.2025.073076.
17. Abdelhaq M, Al-Shamayleh AS, Akhunzada A, Ivković N, Hasan T. Scalable and resilient AI framework for malware detection in software-defined internet of things. Comput Mater Contin. 2026;87(1):1–10. doi:10.32604/cmc.2025.073577.
18. Natsos D, Symeonidis AL. Transformer-based malware detection using process resource utilization metrics. Results Eng. 2025;25:104250. doi:10.1016/j.rineng.2025.104250.
19. Sun CX, Han SH, Shin SS. Malware detection and classification system based on behavior. J Inf Commun Converg Eng. 2025;23(2):121–8. doi:10.56977/jicce.2025.23.2.121.

20. Zhang S, Gao M, Wang L, Xu S, Shao W, Kuang R. A malware-detection method using deep learning to fully extract API sequence features. Electronics. 2025;14(1):167.
21. Nair SJ, Syam SR. Automated malware detection using memory forensics. In: Proceedings of the 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT); 2024 Jun 24–28; Kamand, India. p. 1–5.
22. Maniriho P, Mahmood AN, Chowdhury MJM. MeMalDet: a memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations. Comput Secur. 2024;142(6):103864. doi:10.1016/j.cose.2024.103864.
23. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, et al. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017 Oct 30–Nov 03; Dallas, TX, USA. p. 1175–91.
24. Golmaryami M, Taheri R, Pooranian Z, Shojafar M, Xiao P. SETTI: a self-supervised AdvErsarial malware DeTection ArchiTecture in an IoT environment. ACM Trans Multimedia Comput Commun Appl. 2022;18(2s):1–21. doi:10.1145/3536425.
25. Fang W, He J, Li W, Lan X, Chen Y, Li T, et al. Comprehensive Android malware detection based on federated learning architecture. IEEE Trans Inf Forensics Secur. 2023;18:3977–90. doi:10.1109/tifs.2023.3287395.
26. Li B, Wu Y, Song J, Lu R, Li T, Zhao L. DeepFed: federated deep learning for intrusion detection in industrial cyber–physical systems. IEEE Trans Ind Inf. 2021;17(8):5615–24. doi:10.1109/tii.2020.3023430.
27. Rey V, Sánchez Sánchez PM, Huertas Celdrán A, Bovet G. Federated learning for malware detection in IoT devices. Comput Netw. 2022;204(7):108693. doi:10.1016/j.comnet.2021.108693.
28. Jiang Y, Tong X, Liu Z, Ye H, Tan CW, Lam KY. Efficient federated unlearning with adaptive differential privacy preservation. In: Proceedings of the 2024 IEEE International Conference on Big Data (BigData); 2024 Dec 15–18; Washington, DC, USA. p. 7822–31.