Applied Computing and Informatics

Obfuscated file-less malware detection using integrating memory forensics data with machine learning techniques

Mohamed Zakaria, Mohamed S. Mohamed, Sherif Hussein and Gouda I. Salama

Computer Engineering and A.I. Department, Military Technical College, Cairo, Egypt

Received 24 February 2025
Revised 3 June 2025
24 August 2025
Accepted 12 September 2025

Abstract

Purpose – Malware remains one of the most critical Internet security concerns, with recent advancements making it increasingly sophisticated and capable of evading traditional antivirus protection. Modern threats have shifted from file-based to file-less malware, which resides in memory and bypasses conventional detection mechanisms. This study aims to propose an improved approach for detecting file-less malware families and subfamilies to support effective incident response.

Design/methodology/approach – The proposed approach integrates memory forensics data with machine learning techniques, utilizing ensemble soft voting to enhance detection. The method focuses on reducing detection time at each categorization level, including binary classification, family classification and subfamily classification, while maintaining high prediction accuracy.

Findings – The suggested method achieves 99.9% accuracy for binary classification, 88.86% accuracy for multiclass malware family classification and 77.40% accuracy for multiclass malware subfamily classification. Furthermore, the detection time was significantly reduced: binary classification from 3.359002 to 1.895786 seconds, family classification from 10.190352 to 3.694623 seconds and subfamily classification from 12.114225 to 3.737292 seconds when selected features were applied. These results demonstrate the effectiveness of the proposed strategy in mitigating sophisticated file-less malware.

Originality/value – This research contributes by presenting a novel integration of memory forensics and machine learning with ensemble soft voting to improve both detection accuracy and speed in file-less malware analysis. The approach offers a practical advancement in protecting systems against advanced malicious programs by addressing the challenges of malware classification at binary, family and subfamily levels.

Keywords File-less malware, Malicious software, Cyber security, Detection techniques, Memory forensics

Paper type Research article

1. Introduction and background

The term malware refers to software that is intentionally created to exploit vulnerabilities in computer systems and gain unauthorized access or acquire sensitive personal information [1]. This term was originated in the 1970s with the emergence of the Creeper program [2]. It encompasses a wide range of malicious programs and code that pose significant risks to the security and privacy of computer users. It can come in many forms, including computer viruses, worms, Trojan horses, ransomware, spyware, cryptocurrency miners, adware and other dangerous programs [3].

The motivations behind creating malware can vary. However, a common objective is to gather personal data from the targeted system. This data can be used for various malicious purposes, such as identity theft, financial fraud or espionage.

Conflict of interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Applied Computing and Informatics
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-02-2025-0052

Recently, the malware industry has made significant advancements, changing the entire malware development landscape. Cyber-criminals have become more sophisticated as they move from file-based to file-less malware which became very popular in 2017 when attacks surged by 29% [4, 5]. File-less malware, does not require a traditional file system but instead uses benign processes to achieve its malicious goals.

Malware can be classified into different types or forms, and each type is associated with a specific family. Within each family, there are multiple versions of malware, referred to as subfamilies. These subfamilies share common characteristics, behaviors or code patterns [6].

(1) File-Based Malware

File-Based Malware is delivered by executables, documents or scripts. It can infect the system if obtained from a suspicious website, transferred via an infected flash drive or received as an email attachment. Infected files are usually stored on the hard disk. Users with antivirus software can scan files for malware. After identifying a file as malicious, antivirus software can remove or quarantine it to prevent execution. If antivirus software fails to detect a file as malicious, users may run it, infecting the system with malware [7].

(2) File-less Malware

Attackers have developed file-less malware in a way that presents a significant challenge for antivirus software to detect and counteract. Unlike traditional malware that stores its malicious code on the computer's hard drive, file-less malware directly injects its code into the computer's memory. By exploiting system vulnerabilities or utilizing legitimate system tools, it carries out malicious activities without leaving any traces on the hard drive [8]. This grants the malware persistence on system administration tools such as PowerShell (PS) and Windows Management Instrumentation (WMI), which are commonly employed for executing and distributing file-less malware [9]. Additionally, file-less malware can propagate through phishing attacks that deceive users into clicking on malicious links or downloading harmful files. As a result, the hiding capabilities of file-less malware often render current antivirus techniques ineffective in detecting it [10].

The file-less malware attack life cycle includes four stages: delivery, persistence, execution and objective. Initially, perpetrators use social engineering to secretly download and install the malicious script into the computer's memory to avoid detection. The second stage involves building a lasting presence. Malware can persist after reboots by injecting itself into system processes. The third stage involves file-less malware using legitimate system resources to perform malicious activities discreetly and undetected. In the fourth stage, fileless malware conceals actions and avoids security measures [11]. The absence of traditional file-based traces and the utilization of legitimate resources make it difficult for security measures to identify and mitigate the file-less malware [9].

(3) Handling Obfuscation and Anti-Forensics

Obfuscation techniques such as polymorphism, encryption of payloads and reflective DLL injection are commonly used to evade static analysis. Our approach addresses these by relying on dynamic memory-resident behavior captured in memory dumps. Since these behaviors manifest during execution (e.g., abnormal memory handles, malicious thread creation), our feature vectors retain discriminative power even when code is obfuscated.

(4) Malware Analysis and Detection Techniques

Malware analysis and detection are two distinct but interconnected processes in the field of cybersecurity. Malware analysis involves in-depth examination of malicious software to understand its behavior and impact [12]. It can be divided into two main categories: (1) Static Analysis involves studying code and structure without their execution in order to identify the presence of malware. (2) Dynamic Analysis entails observing behavior during execution

within a controlled environment to identify both known and unknown malicious code [13]. On the other hand, malware detection aims to identify malware or malicious activities within a system or network. There are three fundamental methods for malware detection include signature-based, heuristic and machine learning based detection. The latter one plays a crucial role in detecting non-traditional malware, as it analyzes large datasets, identifies patterns and adapts to evolving threats. Methods such as signature-based detection use predefined patterns which may result in high rates of false positives and negatives [14], while heuristic analysis relies on behavioral rules, for example, in the case of file-less malware, it is utilized to detect malware through the extraction of features from memory analysis using" Volatility" memory forensics framework [3, 15]. However, machine learning based detection excels in identifying complex and emerging threats by learning from labeled samples and detecting anomalies. Its ability to continuously update models enhances the effectiveness of malware detection.

This research presents a novel approach to detect obfuscated malware using machine learning algorithms and utilize an ensemble voting that improves the accuracy of detection by combining multiple classifier models. Additionally, the approach tracks malware families and sub-families, allowing for more effective incident response processes. Various performance measures were studied to verify the efficiency of the proposed approach, with a comparison of performance indicators across a collection of individual ML classifiers. Unlike previous approaches, our method introduces a dual-layer ensemble combining bagging and boosting with soft voting, specifically optimized through a custom feature selection pipeline tailored for obfuscated file-less malware. This contributes both to accuracy (up to 88.86% in family-level classification) and drastic reductions in detection time (up to 69.15%). To our knowledge, this level of combined detection speed and granularity (binary – family – sub-family) has not been previously reported in the literature.

2. Related work

There are a number of works conducted for file-less binary classification in which all proposed methods detect only if the samples are malware or benign. The authors in Ref. [16] present a method is based on the advanced VolMemlyzer tool for extracting memory features in learning systems. It focuses on hidden obfuscated malware in coupled machine learning models. This technique provides an effective framework for malware detection. A specific malware memory dataset (MalMemAnalogy-2022) was created to test and evaluate the framework, simulating real-world obfuscated malware. The results demonstrate that the proposed solution can quickly identify ambiguous and hidden malware by utilizing memory feature engineering. They achieved a detection accuracy up to 99% in binary classification.

Another method in the study [17] utilizes Pyspark on Google Collaboratory's Apache Spark platform to investigate the (CIC-MalMem-2022) balanced dataset, employing various algorithms for binary classification, including Random Forest (RF), Decision Trees (DT), eXtreme Gradient Boosting (Xgboost), Logistic Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron (MLP), Feedforward Deep Neural Network (FDNN) and Short-Term Memory algorithms (STM). They achieved a detection accuracy up to 99.9% in binary classification. The findings suggest that memory analysis data are useful for identifying malware, and deep learning and machine learning techniques trained on memory datasets achieved positive outcomes in malware detection.

Various algorithms were employed in Ref. [18] which including Binary Bat Algorithm (BBA), Cuckoo Search Algorithm (CSA), Mayfly Algorithm (MA), Particle Swarm Optimization (PSO), K-Nearest Neighbor (kNN) and RF. SVM was used to classify and evaluate malicious records using (CIC-MalMem-2022) dataset, and found it effective for malware detection by achieving a detection accuracy up to 99.9% in binary classification. The research offers a model for examining disguised malware in computer memory and illustrates the efficiency of machine learning methods in distinguishing between legitimate data and malware.

A machine learning method with memory forensics is demonstrated in Ref. [19] to examine 33 distinct attributes of in-memory and file-less browser attacks. The suggested technique produced a dataset of file-less malware acquired from the Virus Total, Any Run and PolySwarm platforms achieving a detection accuracy up to 93.33% in binary classification. The suggested technique demonstrates positive outcomes in identifying file-less malware, which is notorious for evading traditional detection tactics.

According to what was previously mentioned, binary classification has helped determine whether the malware is infected, but it did not help contain the cyberattack or infection because the type of infection was not determined. Identifying the malware family and sub-family to which the sample belongs provides valuable insights into the type of infection. Behavioral traits, motives, tactics, spread methods and potential impact on compromised systems help cyber incident response teams understand targeted attacks, track malware evolution and design mitigation strategies for specific families or subfamilies to effectively address the attack.

Recently there are also those who made multiclass classification for detect malware families and sub-families. A method using (CIC-Malmem2022) dataset in Ref. [20] presents to propose a multiclass malware classification and detection method that can identify specific attack types to identify state-of-the-art malware suitable for deployment on embedded devices. They achieved a detection accuracy of up to 99.96% in binary classification, 84.56% in multi-class malware family classification and 72.60% in multi-class malware sub-family classification. The experiments show that this method outperformed prior techniques in detecting Obfuscated Memory Malware (OMM) and identifying attack types. The study provides a detection algorithm for obfuscated malware that works well with limited system requirements. This study is valuable for malware detection in IoT devices, especially in smart city applications.

The authors in Ref. [21] propose a novel hybrid classification model, named MalHyStack, that utilizes ensemble learning and deep learning techniques to detect obfuscated malware within a network using the (CIC-MalMem-2022) dataset, achieving a detection accuracy up to 99.98% in binary classification, 85.04% in multi-class malware family classifications and 70.29% in multi-class malware sub-family classifications. The study examines recent advancements in analyzing hidden malware and emphasizes the effectiveness of combining different classification models in this area. In general, the study proposes a promising method for identifying hidden malware and provides valuable knowledge on the use of hybrid classification models in this field.

Recent research [22] proposes a DNN model that uses ANOVA *F*-test to select the most informative features for improved classification accuracy. Feature score distributions in ANOVA for numerical input data identify the most effective features. The model for binary and multiclass malware classification was tested using CIC-Mal mem-2022. According to experimental data, the model achieved 100% binary classification accuracy, 85.83% family-level detection accuracy and 73.98% individual attack identification accuracy.

Multi-classification has helped mitigate cyberattacks by identifying malware families and sub-families. This aids incident response and targeted mitigation. However, these malware classifications are currently inaccurately detected. Combating such attacks requires improvements.

The key contributions and results of the related work discussed above are outlined in Table 1 for clarity.

3. Methodology: integrating memory forensics with machine learning

A hybrid bagging and boosting ensemble model is proposed with feature selection methodology for analyzing obfuscated malware. Efforts were made to enhance the overall performance of our approach, considering the significance of early detection of malware infection. Consequently, work was undertaken to enhance both the accuracy of detecting and identifying these malwares and the detection time required to determine whether a sample is

Table 1. Summary of state-of-the-art techniques for file-less malware detection

Ref./ year	Dataset used	Method	Accuracy	Feature selection	Limitations
[16]	CIC-MalMem-2022	Stacked Ensemble (NB, RF, DT + LR)	Binary: 99.00%	No	Focused on binary classification only
[17]	CIC-MalMem-2022	Various ML/DL	Binary: 99.97%	Yes	Focused on binary classification only
[18]	CIC-MalMem-2022	FS (BBA, MA, etc.) + RF	Binary: 99.99%	Yes	Focused on binary classification only
[19]	Custom fileless dataset (45 samples)	Volatility features + ML	Binary: 93.33%	No	Focused on binary classification only
[20]	CIC-MalMem-2022	Compact CNN-BiLSTM models	Binary: 99.96% Family: 84.56%, Subfamily: 72.60%	No	Multiclass classification Performance drops
[21]	CIC-MalMem-2022	Hybrid Stack: (ET, XGB, RF) + DL	Binary: 99.98% Family: 85.04% Subfamily: 70.29%	Yes	Multiclass classification Performance drops
[22]	CIC-MalMem-2022	(DNN) with ANOVA *F*-test feature selection	Binary: 99.99% Family: 85.83% Subfamily: 73.98%	Yes	Multiclass classification Performance drops and Computational cost

Source(s): Authors' elaboration

benign or malicious and to identify the family and subfamily of malignant samples. This section is divided into four subsections to provide a detailed description of the proposed approach.

3.1 The proposed architecture

The proposed model architecture as shown in Figure 1 is designed to achieve both binary and multi-class classification objectives. In the binary classification setting, the model discriminates between malicious and benign instances. The approach is extended to multi-class classification to categorize identified malware into specific families and sub-families, providing a more nuanced analysis of threat landscapes.

The proposed model is divided into two phases, the first phase comprises training the model with all machine learning algorithms which are selected from the machine learning models used in the relevant studies in the field of fileless malware analysis and detection to perform evaluation on these models and choose the best three models to be used in the second phase by combining their predication then applying ensemble bagging and boosting with soft voting to get their final prediction.

3.2 Memory Artifact Extraction

We utilize the CIC-MalMem-2022 dataset, which is derived from real memory dumps analyzed through the Volatility memory forensics framework. These artifacts include critical memory-based indicators such as registry key handles, dynamically loaded DLLs, injected threads, suspicious memory allocations and anomalous process behaviors. Volatility plugins (e.g., pslist, dlllist, handles, malfind) were used to extract these indicators from raw memory images. The extracted information was then normalized and encoded into structured feature vectors suitable for machine learning algorithms. These vectors represent high-dimensional behavioral snapshots of memory states, enabling the detection of obfuscated and file-less

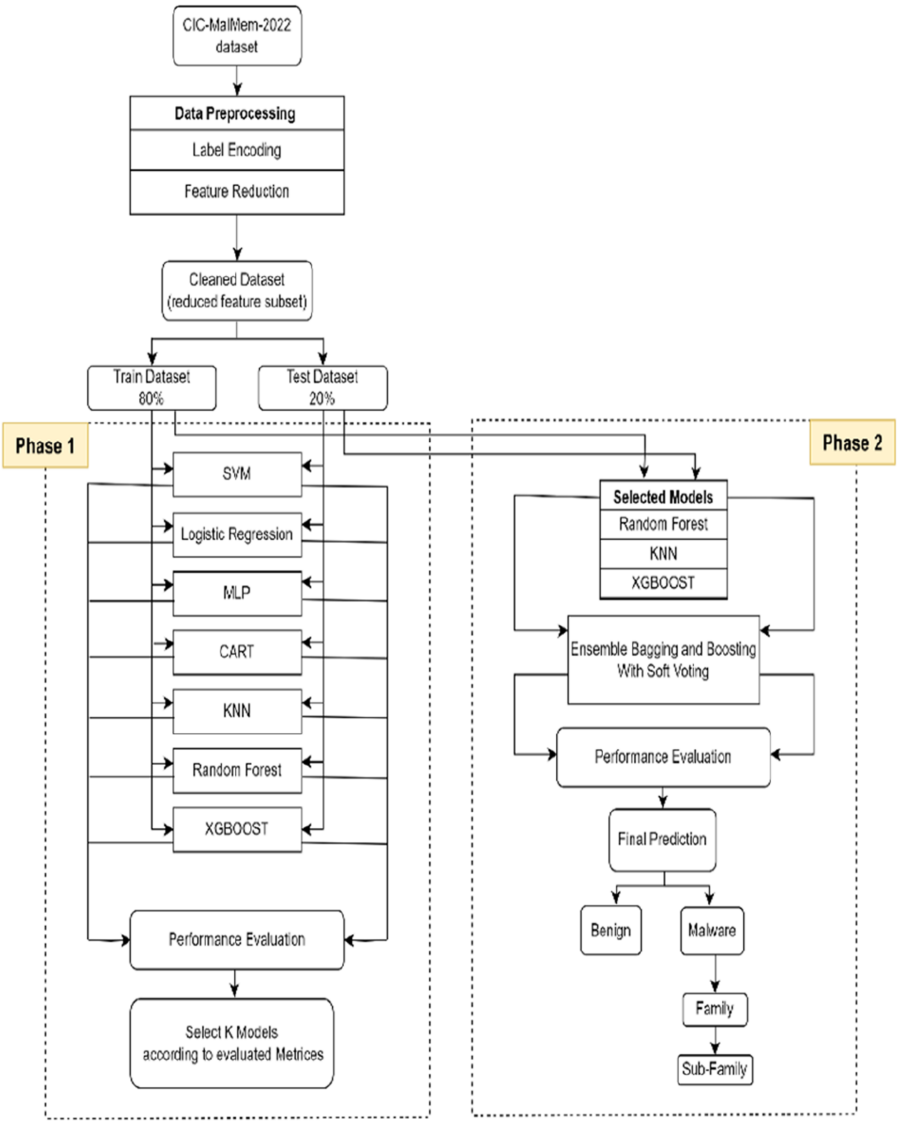

Figure 1. The proposed approach operational flow block diagram. Source: Authors' elaboration

malware based on runtime characteristics rather than static signatures. By leveraging these memory-resident features, our approach can effectively distinguish between benign and malicious activities even when traditional detection methods fail due to encryption, packing or runtime obfuscation.

3.3 Dataset description

The Canadian Institute of Cybersecurity released CIC-MalMem-2022 [23], a dataset of 58,596 disguised malware records comprising 29,298 benign and 29,298 malicious instances. In this

dataset, widely observed malware was used to simulate a real-life scenario. It includes Spyware, Ransomware and Trojan with 15 malware sub-families.

It includes a wide range of features that can be used for malware classification, detection and analysis and according to these features we can understand why something is flagged as malicious or benign.

To ensure that the dataset can be effectively classified, several preprocessing steps are necessary. These steps are crucial for optimizing classification models and preparing the data for use with machine learning algorithms. Additionally, data balancing operations are often necessary to combat overfitting problems, particularly in deep learning approaches. However, the dataset utilized in this study is already balanced, consisting of two classes: benign and malware. As such, it is resistant to overfitting. In this study, categorical values of "Benign" and "Malware" were assigned to 0 and 1 for binary classification. For multiclass classification, malware families were assigned values of 0, 1, 2 and 3, and for sub-families, values of 0 through 15 were assigned. These assignments were made to prepare the data for use with machine learning and deep learning algorithms.

3.4 Feature selection and dimensions reduction

(1) Feature Selection

The principal emphasis of the research revolved around improving the comprehensive detection rate by focusing on improving the accuracy and reducing the detection time. In order to improve the performance of the proposed model, four feature selection and dimension reduction techniques are used: Mutual Information (MINFO), Principal Component Analysis (PCA), Analysis of Variance (ANOVA) and Chi-Square Test (CHI^2).

The statistical metric MINFO measures how much one variable can teach another, indicating mutual dependence. MINFO eliminates all but the most useful target variable features (class label). This useful method captures linear and non-linear feature-target variable relationships [24].

PCA selects and reduces features while preserving key patterns and relationships [25].

ANOVA helps choose by highlighting significant differences between classes or categories. Use each feature's F-statistic to calculate class variance ratios. Class distinction improves with higher F-statistic features [26].

CHI^2 selects features based on feature-class independence. Classification benefits from high chi-square scores, which indicate stronger target variable relationships [27].

Applying these methodologies allows for selecting the least features while maintaining model accuracy. Using these techniques. We applied it to the three models eXtreme Gradient Boosting (Xgboost), RF and kNN that we worked on, and selected the number of components from each model that could maintain the same accuracy while improving performance.

In case of Binary Classification, the least number of features that can used is 39 features for Xgboost by using PCA technique, 25 features for RF by using the ANOVA technique and 14 features for kNN by using the PCA technique.

In case of Family Classification, the least number of features that can used is 33 features for Xgboost by using the MINFO technique, 34 features for RF by using the ANOVA technique and 12 features for kNN by using the MINFO technique.

For Sub-Family Classification, the least number of features that can used is 48 features for Xgboost by using the MINFO technique, 39 features for RF by using the MINFO technique and 11 features for kNN by using the MINFO technique.

(2) Hyperparameter Tuning and Cross-Validation

Hyperparameters were optimized using GASearchCV with 10-fold stratified cross-validation, focusing on F1-score optimization. For example, RF used 100 estimators, no depth limit; Xgboost used a learning rate of 0.1, max depth of 6 and 100 trees.

3.5 Ensemble learing techniques

(1) Individual model training

To guarantee the efficacy of our model, we utilized seven distinct machine learning algorithms: RF, SVM, Classification and Regression Trees (CART), MLP, kNN, LR and Xgboost. This enabled us to assess the performance of each algorithm and determine the most appropriate model for detecting file-less malware.

(2) Using Voting Classifier

We utilize a voting classifier with bagging and boosting techniques, which combined the predictions of our top three models kNN–Xgboost–RF which achieved the highest performance during the three stages of classification (Binary, Family and Sub-Family). The voting mechanism was evenly weighted to ensure equal importance among the models. Additionally, we opted for a soft-voting strategy that allowed for probabilistic outputs, resulting in a more precise decision-making process. Bagging is a technique aimed at reducing variance and improving generalization. In the bagging process, we initialize a bagged model B containing N_{bags} fitted base models and train these multiple base models on bootstrap samples of the training data as shown in Equation (1).

$$B = \left\{M_1, M_2, \ldots, M_{N_{\text{bags}}}\right\} \tag{1}$$

M_i represents the base estimator or the base model that is used as the building block for creating the ensemble models. It can be any machine learning algorithm such as kNN, RF and Xgboost. On the other hand, boosting creates a strong learner by iteratively combining multiple weak learners. The boosted model A is initialized with Nboost iterations fitted models, their coefficients and iteratively train base models. Then assigning higher weights to misclassified samples as shown in Equation (2).

$$A = \left\{(M_1, a_1), (M_2, a_2), \ldots, \left(M_{N_{\text{boost\_iterations}}}, a_{N_{\text{boost\_iterations}}}\right)\right\} \tag{2}$$

(M_i,a_i) indicates a specific weak learner Model and its corresponding weight.

In order to predict a sample x, the predictions p_B from model B are obtained by averaging over multiple bags, as shown in Equation (3). Similarly, the predictions p_A from model A are calculated using a weighted sum of the boosting iterations, as detailed in Equation (4). Finally, we combine predictions from both bagged and boosted models using ensemble soft voting and the normalized average of these predictions, $p(x)$, is computed as described in Equation (5).

Step 1. Get predictions p_B *from* B

$$p_B(x) = \frac{1}{N_{\text{bags}}} \sum_{i=1}^{N_{\text{bags}}} M_i(x) \tag{3}$$

Step 2. Get predictions p_A from A

$$p_A(x) = \frac{\sum_{j=1}^{N_{\text{boost\_iterations}}} a_j \cdot M_j(x)}{\sum_{j=1}^{N_{\text{boost\_iterations}}} a_j} \tag{4}$$

Step 3. Return normalized average $p(x)$

$$p(x) = \frac{p_B(x) + p_A(x)}{2} \tag{5}$$

4. Experimental results and analysis

All tasks related to this paper, utilized on HP Pavilion 15-dk1009ne, Intel(R) Core (TM) i5-10300H, 2.50GHz CPU, 8 GB RAM and Python 3.11 using Jupiter platform.

For a robust model evaluation in our experiment, we employed 10-fold cross validation and an 80/20 stratified train-test split for a robust model evaluation in our experiment. A genetic algorithm (GASearchCV) assisted in optimizing hyperparameters over 20 generations with a 10-person population targeting the best configuration for each model. The optimized settings for the classifiers are as follows:

(1) RF: number of trees = 100, max depth = None, min samples split = 2.

(2) SVM: kernel = RBF, C = 1.0, γ = scale.

(3) CART: max depth = None, criterion = gini, min samples leaf = 1.

(4) MLP: hidden layers = (100), activation = ReLU, solver = adam, max_iter = 200.

(5) kNN: number of neighbors k = 5, metric = Euclidean.

(6) LR: penalty = l2, solver = liblinear; C = 1.0.

(7) Xgboost: learning rate = 0.1, max depth = 6, n_estimators = 100, subsample = 0.8.

These parameters were either selected via genetic search or retained at standard defaults when no significant improvement was observed. Finally, a soft voting ensemble was applied, combining bagging and boosting strategies to evaluate the final performance of the top models.

To evaluate our proposed approach, we used four performance metrics: Accuracy (6), Recall (7), precision (8) and F1-Score (9). We also compare the detection time of our proposed model before and after using the features selection and dimension reduction techniques to measure the improvement in the detection time (10) to enhance the overall performance of our model.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{F1} - \text{Score} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right) \tag{9}$$

$$\text{Detection time Improvement Ratio } (\%) = \frac{\text{Detection Time (Before)} - \text{Detection Time (After)}}{\text{Detection Time (Before)}} \times 100 \tag{10}$$

4.1 Evaluating Individual Model

We utilized seven machine learning algorithms to compare their performance and determine the best suitable model for detecting file-less malware.

(1) Binary Classification

The performance of the binary classification models, that only determines whether a sample is malware or benign, is shown in Table 2. The results indicate that all models have very high and comparable performance, with precision, recall, F1-score and accuracy all exceeding 0.92 for most models. The highest performing models appear to be Xgboost, kNN and RF, with marginally better scores across the metrics compared to the other models. However, the differences are relatively small, and all the models demonstrate excellent classification performance for this task.

(2) Malware Family and Sub-family classification

Compared to the previous malware binary classification, the performance of the models appears to be slightly lower across the metrics. Again, the best performing models seem to be Xgboost, RF and kNN, with relatively higher precision, recall, F1-score and accuracy compared to the other models. CART also show good performance, while SVM, MLP and LR have the lowest scores among the evaluated models. This indicates that the choice of model can have a significant impact on the classification accuracy for this specific problem.

These models can naturally deal with interactions between features, which is crucial in malware detection tasks, especially when moving from binary to multi-class classification. The data set contains imbalanced classes and nonlinear relationships. Therefore, it is best to deal with imbalanced classes and nonlinear relationships through these three models. These three models are the top performers in detecting malware families as shown in Table 3.

Table 2. Performance metrics for binary classification

Model	Precision	Recall	F1	Accuracy
SVM	0.999659	0.999659	0.999659	0.999659
RF	0.999829	0.999829	0.999829	0.999829
CART	0.997782	0.997782	0.997782	0.997782
KNN	0.998891	0.998891	0.998891	0.998891
XGBOOST	0.999829	0.999829	0.999829	0.999829
MLP	0.925512	0.925512	0.924987	0.925512
LR	0.998635	0.998635	0.998635	0.998635

Source(s): Authors' elaboration

Table 3. Performance metrics for family classification

Model	Precision	Recall	F1	Accuracy
SVM	0.757935	0.757935	0.751261	0.757935
RF	0.880290	0.880290	0.880326	0.880290
CART	0.800085	0.800085	0.800034	0.800085
KNN	0.814590	0.814590	0.814589	0.814590
XGBOOST	0.886007	0.886007	0.885957	0.886007
MLP	0.702304	0.702304	0.668437	0.702304
LR	0.720137	0.720137	0.719448	0.720137

Source(s): Authors' elaboration

In malware sub-family classification, models perform significantly lower across all metrics. Malware sub-family classification results in greater difficulty than family-level classification. Xgboost appeared to be the most effective model among the three evaluated models. Although RF and kNN perform well, CART, MLP and LR have the lowest scores as shown in Table 4.

4.2 Ensemble Technique

An ensemble voting strategy improves predictive accuracy. We chose Xgboost, RF and kNN as our main models because of their high binary and multi-class classification performance according to section 3.4. After training these models individually, we used bagging and boosting ensemble techniques to improve performance. To create a robust ensemble model, we used soft voting to average these bagged and boosted models' predicted probabilities. As illustrated in Tables 5–7, this approach uses each method's strengths to improve binary and multi-class classification accuracy and robustness against the individual models.

4.3 Improve malware detection time

The primary focus of the study was on enhancing the overall rate of detection. To improve the model's performance, four techniques for feature selection and dimension reduction were

Table 4. Performance metrics for sub-family classification

Model	Precision	Recall	F1	Accuracy
SVM	0.641724	0.641724	0.639668	0.641724
RF	0.764164	0.764164	0.762391	0.764164
CART	0.620222	0.620222	0.606767	0.620222
KNN	0.676024	0.676024	0.676406	0.676024
XGBOOST	0.774744	0.774744	0.771555	0.774744
MLP	0.593259	0.593259	0.580081	0.593259
LR	0.592577	0.592577	0.576940	0.592577

Source(s): Authors' elaboration

Table 5. Performance metrics for ensemble binary classification

Model	Precision	Recall	F1	Accuracy
RF	0.9998293	0.9998293	0.9998293	0.9998293
KNN	0.9988907	0.9988907	0.9988907	0.9988907
XGBOOST	0.9998293	0.9998293	0.9998293	0.9998293
Proposed ensemble approach	0.9998293	0.9998293	0.9998293	0.9998293

Source(s): Authors' elaboration

Table 6. Performance metrics for ensemble family classification

Model	Precision	Recall	F1	Accuracy
RF	0.8799488	0.87994880	0.88000401	0.87994880
KNN	0.8145904	0.81459044	0.81458906	0.81459044
XGBOOST	0.8860068	0.88600682	0.88595725	0.88600682
Proposed ensemble approach	0.8881399	0.88813993	0.88813337	0.88813993

Source(s): Authors' elaboration

Table 7. Performance metrics for ensemble sub-family classification

Model	Precision	Recall	F1	Accuracy
RF	0.7651023	0.7651023	0.7636136	0.765102
KNN	0.6760238	0.6760238	0.6764061	0.676023
XGBOOST	0.7747440	0.7747440	0.7715545	0.774744
Proposed ensemble approach	0.7768771	0.7768771	0.7754447	0.776877

Source(s): Authors' elaboration

utilized: MINFO, PCA, ANOVA and CHI^2. These techniques were applied to the three models Xgboost, RF and kNN, and the number of components from each model that could maintain the same accuracy while improving performance was selected.

Finally, the proposed bagging and boosting with soft voting model was implemented, and an improvement in detection time (the time taken to detect whether a sample is benign or malicious and to identify the family and subfamily of malignant samples) was observed for each stage of classification (binary–family–subfamily), as illustrated in Figure 2. The detection time was (3.359002) sec for Binary classification, (10.190352) sec for Family classification and (12.114225) sec for Sub-Family classification. After the selected features were utilized, the detection time decreased to (1.895786) sec for Binary classification, (3.694623) sec for Family classification and (3.737292) sec for Sub-Family classification.

Due to the increased complexity and number of classes, malware family and sub-family classification models perform slightly worse than binary classification. However, the Xgboost, RF and kNN ensemble model performed well. We calculated macro-average and micro-average precision, recall and F1-score to assess classification performance. The malware family classification model had macro-precision of 0.887, macro-recall of 0.886 and macro-F1-score of 0.885, and micro-precision, recall and F1-score of 0.888. These close values show that the model works across all family classes, regardless of distribution.

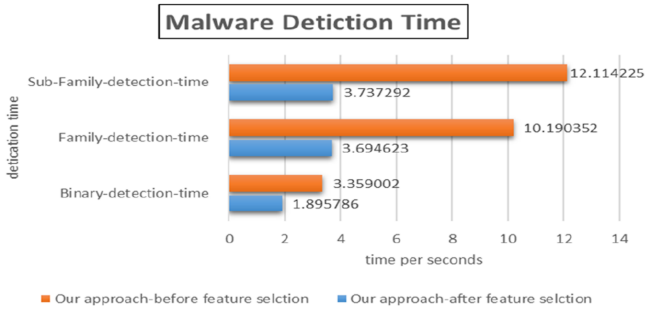

Figure 2. Proposed approach detection time improvement. Source: Authors' elaboration

Table 8. Detection time improvement ratio

Study	Binary detection time	Family detection time	Sub-family detection time
Proposed method-before feature selection	3.359002 sec	10.190352 sec	12.114225 sec
Proposed method-after feature selection	1.895786 sec	3.694623 sec	3.737292 sec
Improvement ratio	(43.56)%	(63.74)%	(69.15)%

Source(s): Authors' elaboration

Table 9. Comparison of malware classification results and improvement ratios

Study	Binary classification				Multi-class classification							
					Family				Sub-family			
	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
[20]	1.0	1.0	1.0	0.9996	0.85	0.85	0.84	0.845	0.73	0.73	0.72	0.726
[21]	0.999	0.999	0.999	0.999	0.850	0.851	0.849	0.850	0.702	0.702	0.703	0.702
[22]	0.999	0.999	0.999	1.0	0.86	0.86	0.86	0.858	0.74	0.75	0.74	0.739
Proposed approach	0.999	0.999	0.999	0.999	0.886	0.886	0.886	0.886	0.774	0.774	0.774	0.774
Improvement ratio	–-				(3.27–4.82)%				(4.63–10.13)%			

Source(s): Authors' elaboration

Results are competitive for 16-subfamily classification. Micro-precision, recall and F1-score were 0.774, macro-precision, recall and macro-F1-score were 0.772, 0.771 and 0.770. It appears that the model balances detection with fine-grained class distinctions. Similar micro and macro metrics show that the classifier does not favor any subfamily class, confirming its generalization across categories.

5. Results analysis

Detecting file-less malware is a relatively new field in cyber security. Consequently, we can only compare the proposed approach detection accuracy to a similar approach from the existing literature in Refs. [20–22]. They achieved a detection accuracy of up to 99.96%, 99.98% and 100% in binary classification, achieved a detection accuracy of up to 84.56%, 85.04% and 85.83% in multi-class malware family classification and finally, they achieved a detection accuracy of up to 72.60%, 70.29% and 73.98% in multi-class malware sub-family classification as shown in Table 9.

However, we are able to obtain a detection accuracy of up to s99.99% in binary classification, 88.81% in multi-class malware family classification and 77.68% in multiclass malware sub-family classification. Efforts were dedicated to enhancing the overall performance of our approach, with a focus on the crucial aspect of early detection of malware infection. Subsequently, improvements were pursued in the detection time.

The proposed approach achieved improvement in malware detection time up to 43.56% in binary classification, 63.74% in multi-class malware family classification and 69.15% in multi-class malware sub-family classification as depicted in Table 8. The suggested approach makes use of machine learning algorithms in combination with a bagging and boosting ensemble learner employing soft voting, which exhibits superior classification accuracy as compared to relevant studies as outlined in Table 6.

6. Conclusion

The proposed approach takes into account both binary and multiclass classification for detecting malware, encompassing both malware family and malware sub-family. It achieves accuracy rates of up to 99.99%, 88.86% and 77.40% in binary, family and subfamily classification, respectively. Despite its apparently modest scale compared to relevant studies, this advancement carries significant weight in the realm of cybersecurity. Even a marginal increase of around 6.6% in detection accuracy for subfamily classification can have noteworthy implications, fortifying overall security measures. Conversely, after feature selection is applied there is an enhancement in detection time for each classification stage of up to 43.56%, 63.74% and 69.15% in binary, family and subfamily classifications, respectively. This time enhancement is substantial, considering the criticality of identifying malicious software early on to safeguard computers against malware and cyber-attacks. In the future work we will integrate the suggested approach with the deep learning-based methods and integrate the suggested approach with the existing threat intelligence platforms to support automated incident response systems and offer actionable insights. The proposed model's rapid detection speed makes it suitable for near real-time endpoint detection systems (EDRs). Integrating it with threat intelligence platforms could automate early-stage triage and threat classification. Future work may focus on transferring this pipeline into live memory scanning environments or adapting it for IoT/edge devices with limited resources.

References

1. Mira F. A systematic literature review on malware analysis. In: Proceedings 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS); 2021. p. 1-6.

2. Kaspersky A brief history of computer viruses and what the future holds; 2023. Available from: https://me-en.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds [accessed 15 March 2024].
3. Sihwail R, Omar K, Ariffin KAZ. A survey on malware analysis techniques: static, dynamic, hybrid and memory analysis. Int J Adv Sci Eng Inf Technol. 2018; 8: 1662-71.
4. CrowdStrike CrowdStrike 2023 global threat report: resilient businesses fight relentless adversaries; 2023. Available from: https://www.crowdstrike.com/blog/global-threat-report-preview-2023/ [accessed 22 July 2024].
5. Ponemon Institute New Ponemon institute study: key findings the 2017 state of endpoint; 2017. Available from: https://www.ponemon.org/newsupdates/blog/security/the-2017-state-of-endpoint-security-risk-report.html [accessed 8 January 2024].
6. Djenna A, Bouridane A, Rubab S, Marou IM. Artificial intelligence-based malware detection, analysis, and mitigation. Sym. 2023; 15(3): 677. doi: 10.3390/sym15030677.
7. Snow D. Investigating fileless malware. Utica College; 2021. Doctoral dissertation.
8. Smelcer J. Rise of fileless malware. Utica College; 2017. Doctoral dissertation.
9. Sanjay BN, Rakshith DC, Akash RB, Hegde VV. An approach to detect fileless malware and defend its evasive mechanisms. In: Proceedings 3rd IEEE International Conference on Computational Systems and Information Technology for Sustainable Solutions; 2018. p. 1-6.
10. Afreen A, Aslam M, Ahmed S. Analysis of fileless malware and its evasive behavior. In: Proceedings 2020 International Conference on Cyber Warfare and Security (ICCWS); 2020. p. 1-6.
11. Tancio B. Hunting for ghosts in fileless attacks, SANS Institute; 2023. Available from: https://www.sans.org/white-papers/38960/ [accessed 30 April 2024].
12. Mohanta A, Saldanha A. Malware analysis and detection engineering. Berkeley, CA: Apress; 2020.
13. Verma A, Rao MS, Gupta AK, Jeberson W, Singh V, A literature review on malware and its analysis; 2013. Available from: https://www.researchgate.net/publication/269399065_A_LITERATURE_REVIEW_ON_MALWARE_AND_ITS_ANALYSI [accessed 12 September 2024].
14. Baskaran B, Ralescu A. A study of android malware detection techniques and machine learning. In: Proceedings 27th Modern Artificial Intelligence and Cognitive Science Conference; 2016. p. 15-23.
15. Volatility Foundation An advanced memory forensics framework(2023). Available from: https://github.com/volatilityfoundation/volatility [accessed 5 June 2024].
16. Carrier T, Victor P, Tekeoglu A, Lashkari A. Detecting obfuscated malware using memory feature engineering. In: Proceedings 8th International Conference on Information Systems Security and Privacy (ICISSP); 2022. p. 1-8.
17. Dener M, Ok G, Orman A. Malware detection using memory analysis data in big data environment. Appl Sci. 2022; 12(17): 8604. doi: 10.3390/app12178604.
18. Ghazi MR, Raghava NS. Machine learning based obfuscated malware detection in the cloud environment with nature-inspired feature selection. In: Proceedings 5th International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT); 2022. p. 1-6.
19. Khalid O, Ullah S, Ahmad T, Saeed S, Alabbad DA, Aslam M, Buriro A, Ahmad R. An insight into the machine-learning-based fileless malware detection. Sensors. 2023; 23(2): 612. doi: 10.3390/s23020612.
20. Shafin SS, Karmakar G, Mareels I. Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. Sensors. 2023; 23(11): 5348. doi: 10.3390/s23115348.
21. Roy KS, Ahmed T, Biswas Udas P, Karim ME, Majumdar S. MalHyStack: a hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intell Syst Appl. 2023; 20: 200283. doi: 10.1016/j.iswa.2023.200283.

22. Hadjila H, Merzoug M, Ferhi W, Moussaoui D, Bouidaine AB, Hachemi MH. Obfuscated malware detection using deep neural network with ANOVA feature selection on CIC-MalMem-2022 dataset. Sci Tech J Inf Technol Mech Opt. 2024; 24(5): 849-57. doi: 10.17586/2226-1494-2024-24-5-849-857.

23. Canadian Institute Canadian Institute for cybersecurity, CIC-MalMem-2022 dataset; 2022. Available from: https://www.unb.ca/cic/datasets/malmem-2022.html [accessed 18 November 2024].

24. Peng H, Long F, Ding C. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and mi redundancy. IEEE Trans Pattern Anal Mach Intell. 2005; 27 (8): 1226-38. Available from: https://doi.org/10.1109/TPAMI.2005.159 [accessed 1 June 2025].

25. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philos Trans R Soc A: Math Phys Eng Sci. 2016; 374(2065): 20150202. doi: 10.1098/rsta.2015.0202.

26. Henson RN. Analysis of variance (ANOVA). In Brain mapping: an encyclopedic reference, Toga AW, Ed. vol. 1. Academic Press, 2015, pp. 477–81. Available from: https://www.mrc-cbu.cam.ac.uk/wp-content/uploads/www/sites/3/2015/03/Henson_EN_15_ANOVA.pdf [accessed 1 June 2025].

27. McHugh ML. The chi-square test of independence. Biochemia Med. 2013; 23(2): 143-9. Available from: https://doi.org/10.11613/BM.2013.018 [accessed 1 June 2025].

Corresponding author
Mohamed Zakaria can be contacted at: mohamed.zakaria1991@icloud.com