SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 2, 2024, 264-276
DOI: 10.35377/saucis...1489237

RESEARCH ARTICLE

Detection and Analysis of Malicious Software Using Machine Learning Models

Ahmet Öztürk[1], Selman Hızal[1]

[1]Sakarya University of Applied Sciences, Department of Computer Engineering, Sakarya, Türkiye

Corresponding author:
Yerniyaz Bakhytov, Sakarya University of Applied Sciences, Department of Computer Engineering, Sakarya, Türkiye
ozturkahmet89@gmail.com

Article History:
Received: 24.05.2024
Accepted: 19.08.2024
Published Online: 26.08.2024

ABSTRACT
The continuous evolution of malware poses a significant challenge in cybersecurity, adapting to technological advancements despite implemented security measures. This paper introduces an innovative approach to enhance the detection of obfuscated malware through the integration of machine learning (ML). Utilizing a real-world dataset of prevalent malware types such as spyware, ransomware, and trojan horses, our study addresses the evolving challenges of cybersecurity. In this study, we evaluate the performance of ML algorithms for obfuscated malware detection using the CIC-MalMem-2022 dataset. Our analysis encompasses binary and multi-class classification tasks under various experimental conditions, including percentage splits and 10-fold cross-validation. The evaluated algorithms include Random Tree (RT), Random Forest (RF), J-48 (C4.5), Naive Bayes (NB), and XGBoost. Experimental results demonstrate the effectiveness of RF, J-48, and XGBoost in achieving high accuracy rates across different classification tasks. NB also shows competitive performance but faces challenges in handling imbalanced datasets and multi-class classification. Our findings highlight the importance of employing advanced ML techniques for enhancing obfuscated malware detection capabilities and provide valuable insights for cybersecurity practitioners and researchers. Future research directions include fine-tuning model hyperparameters, exploring ensemble learning approaches, and expanding evaluation to diverse datasets and real-world scenarios.

Keywords: Information security, Software analysis, Malware detection system, Machine learning

1. Introduction

The persistent evolution of malicious code, or malware, presents significant threats to internet-connected devices despite robust security measures. Malware, designed to perform unauthorized actions often without the user's knowledge, is utilized for malicious purposes such as stealing passwords, accessing confidential data, or corrupting system operations [1]. This poses profound challenges to the core aspects of information security: privacy, integrity, and availability [2]. Malware can expose sensitive organizational data (privacy), alter or corrupt records (integrity), and disrupt system functionality (availability) by deleting or overwriting files or damaging storage media. These characteristics make malware detection difficult, especially as it continually develops techniques to evade conventional detection methods [3].

Malware encompasses various forms, including worms, viruses, bots, trojan horses, ransomware, spyware, adware, spam, phishing, and rootkits, necessitating a comprehensive and nuanced detection strategy [4][5][6][7]. Traditional manual detection methods, while thorough, are impractically time-consuming and complex. This has driven the adoption of automated systems such as Machine Learning (ML) [8]. These intelligent systems can swiftly and accurately analyze data, learning from selected training datasets to optimize detection processes [9]. ML algorithms, varying in speed, accuracy, and precision, significantly impact these systems' outcomes [10]. For instance, ensemble ML techniques are well-suited to malware detection and characterization objectives [11]. By training on labeled datasets, supervised learning algorithms can achieve high accuracy rates in identifying certain types of malware [12]. In contrast, unsupervised learning algorithms can be effective in detecting unknown threats by discovering previously unidentified patterns and relationships in datasets [13]. Additionally, reinforcement learning and deep learning techniques can offer fast and flexible detection mechanisms in complex and dynamic threat environments [14]. In this context, ML-based systems go beyond traditional methods, allowing malware to be detected more effectively and efficiently [15].

Current methods for detecting malware, often involving detailed and slow analysis of computer memory, are impractical for real-world applications. There is a critical need for more efficient and effective solutions. Our proposed approach leverages features identified through memory analysis to enhance malware detection systems [16].

In this study, we use malware samples from distinct categories, such as ransomware, spyware, and trojan horses, to demonstrate our methodological approach. Specifically, we investigate the application of ML models to detect obfuscated malware using the CIC-MalMem-2022 dataset [1]. We conduct binary classification, determining the presence of malware, and multiclass classification, identifying specific malware families. We evaluate a range of traditional ML algorithms, including Random Tree (RT), Random Forest (RF), J-48, Naive Bayes (NB), and XGBoost.

The contribution of the paper can be summarized as follows:

- Evaluating the effectiveness of various machine learning algorithms against obfuscated malware using the CIC-MalMem-2022 dataset.
- Developing a machine learning-based system to address the risks posed by obfuscated malware.
- Providing a comprehensive categorization of the malware families within the dataset.

The remainder of this paper is structured as follows: Section II reviews recent works on obfuscated malware detection, Section III discusses the relevant background, Section IV presents the methodology, Section V details the experimental results, and the final section offers concluding remarks.

2. Related Works

Machine Learning (ML) has become pivotal in enhancing the detection of obfuscated malware within cybersecurity systems [1][17][18][19]. This section reviews several key contributions utilizing ML and Deep Learning (DL) techniques to address the challenges posed by sophisticated malware variants.

Yihan et al. [3] introduced a DL-powered hierarchical model to tackle data imbalance in malware datasets, achieving a binary classification accuracy of 99.67% with the CIC-MalMem-2022 dataset. Ghazi and Raghava [4] applied nature-inspired algorithms for feature selection, notably improving classification accuracy with ML techniques. Their approach demonstrated substantial efficacy in detecting malware in cloud networks, achieving a binary classification accuracy of 99.99% using the Mayfly Algorithm. Nugraha and Zeniarja [11] utilized a Decision Tree (DT) based algorithm for memory-based malware detection, achieving an accuracy of 99.982% in binary classification, with a notable reduction in false positives, highlighting the effectiveness of memory analysis in detecting malware behavior. Dener et al. [16] emphasized the importance of memory analysis in malware detection, employing various ML algorithms with Logistic Regression (LR), showing the highest accuracy of 99.97% in their studies. Tidjon and Khomh [20] proposed using topological data analysis to efficiently identify complex malware patterns, suggesting that TDA techniques combined with ML could enhance the robustness and performance of malware detection systems. Naeem et al. [21] developed a dynamic method for malware detection and classification using anti-analysis techniques, transforming memory dumps into grayscale images. Their hybrid model, combining Convolutional Neural Networks (CNNs) and a Multilayer Perceptron (MLP), achieved high accuracy on Windows (99.1%), Android (94.3%), and Windows obfuscated malware (99.8%). The study suggests further enhancements in training efficiency and resilience against adversarial attacks. Al-Qudah et al. [22] introduced a model combining One-Class Support Vector Machine (SVM) with Principal Component Analysis (PCA), achieving a significant improvement in the detection of memory dump malware, with an accuracy of 99.4% for binary classification. Smith et al. [23] the use of ML for malware detection, employing a range of algorithms to achieve 99.99% accuracy for binary classification, suggesting further research into feature selection and genetic algorithms to enhance detection capabilities.

Mezina and Burget [24] tackled the challenge of detecting obfuscated malware in memory using AI. They utilized the CIC-MalMem-2022 dataset and explored several ML techniques, including Random Forest (RF) and a newly proposed dilated CNN. The RF method achieved an accuracy of 99.99% for binary classification, while the dilated CNN excelled in classifying malware families with an accuracy of 83.53% for multiclass(4) classification. The study highlights the need for improved classification methods to keep up with cybersecurity threats. Roy et al. [25] created MalHyStack, a combined model that uses group learning methods to detect hidden malware very accurately. They stress the importance of detecting these threats early in cybersecurity. They achieved 99.98% accuracy for binary classification, 85.04% accuracy for multiclass(4) classification, and 70.29% accuracy for multiclass(16) classification.

The existing literature often focuses on binary classification, categorizing objects as benign or malware. For instance, Yihan et al. [3] achieved a 99.67% accuracy using a DL-powered hierarchical model, while Ghazi and Raghava [4] attained a 99.99% accuracy with the Mayfly Algorithm for feature selection. Nugraha and Zeniarja's [11] DT-based approach yielded a 99.982% accuracy, and Dener et al. [16] demonstrated a 99.97% accuracy using Logistic Regression. These studies underscore the efficacy of binary classification techniques in identifying malware.

However, this method is insufficient for comprehensive malware analysis. Therefore, there is a need to develop methods that can identify sub-categories of malware. Recent studies have achieved multi-class classification into four groups: benign, ransomware, spyware, and trojan horse attacks. Other studies have gone further, classifying these into 16 sub-types of attacks. Despite these advances, there is still a need for improved results in the literature.

In our research, we addressed this limitation by implementing multilevel classification, categorizing malware into more granular groups. Specifically, we performed classification into 4 main categories (benign, ransomware, spyware, and trojan horse) and further into 16 sub-categories. This approach not only enhances the granularity of malware detection but also improves the ability to identify and respond to new and emerging types of malware. Enhancing these methods could make it easier to detect new and emerging types of malware automatically.

By advancing beyond binary classification to multilevel classification, our study contributes to the field by providing a more detailed and nuanced understanding of malware, which is critical for developing robust malware detection systems. This method allows for more precise identification of malware types and can significantly aid in the development of targeted defense mechanisms against specific types of malware threats.

3. Background

In this section, we provide a contextual overview of malware types with their analysis and detection methods, focusing on the importance of memory analysis in identifying obfuscated malware. Leveraging ML algorithms is highlighted as a key strategy for effectively detecting and mitigating sophisticated malware threats. We then introduce the CIC-MalMem-2022 dataset, a valuable resource extensively employed in malware analysis. Notable for its representation of real-world obfuscated malware scenarios, the dataset's characteristics are outlined, emphasizing its relevance in evaluating ML models for malware detection. Additionally, we discuss the diverse range of ML algorithms utilized to detect obfuscated malware using the CIC-MalMem-2022 dataset and tailored feature extraction methods to enhance detection accuracy. Finally, we present performance metrics utilized for evaluating the effectiveness of ML models, providing quantitative insights into their detection capabilities.

3.1. Overview of Malware Types

In cybersecurity, understanding malware types is crucial for developing effective defense strategies against evolving cyber threats. This study discusses 3 different malware type categories: spyware, ransomware, and trojan horse. Ransomware, with variants such as Ako, Conti, and Maze, poses a significant threat by encrypting files or systems and demanding ransom payments. Spyware, including programs like 180solutions and Gator, compromises privacy and security by secretly monitoring user activity. Trojans like Emotet and Zeus camouflage themselves as legitimate software to facilitate unauthorized access and data leakage. Recognizing the characteristics and tactics of these malware strains allows cybersecurity practitioners to implement robust defenses and effectively mitigate risks.

3.2. Obfuscated Malware Analysis

Obfuscated malware presents a formidable challenge to traditional cybersecurity defenses due to its sophisticated evasion techniques to conceal malicious intent. These evasion strategies include code obfuscation, polymorphism, and encryption, which render malware payloads virtually undetectable to conventional signature-based detection systems. By disguising malicious code within legitimate-looking programs, obfuscated malware evades detection by antivirus software and intrusion detection systems, posing significant threats to individual users and organizations.

In response to the escalating threat posed by obfuscated malware, researchers have turned to advanced detection methodologies, with ML emerging as a promising approach. ML algorithms can discern subtle patterns and anomalies indicative of obfuscated malware within memory dumps. Leveraging labeled datasets such as the CIC-MalMem-2022, researchers can train ML models to recognize obfuscated malware behaviors and distinguish them from benign software. However, the dynamic nature of obfuscated malware necessitates continual adaptation and refinement of detection strategies to combat evolving threats effectively.

3.3. Dataset Description

Canadian Institute for Cybersecurity (CIC) provides data sets for studies in cyber security. The CIC-MalMem-2022 dataset is the most up-to-date dataset created in the malware category by Carrier et al. [1]. This dataset contains a modern collection of samples encompassing obfuscated malware and benign classes. The dataset included prominent malware families such as ransomware, spyware, and trojan horses to simulate real-world conditions. Within the scope of the research, 100 to 200 malware samples were collected from three main categories of trojan, ransomware, and spyware, and five subcategories under each of these categories, as shown in Figure 1.

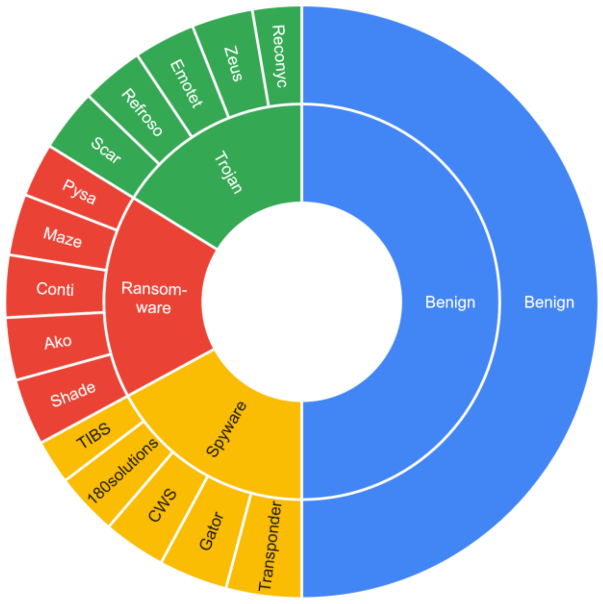

Figure 1. The CIC-MalMem-2022 Dataset Distribution

The CIC-MalMem-2022 dataset creation process stages are shown in Figure 2. The first stage focused on memory dumping, employing VirtualBox as a sandbox to capture memory snapshots from a Microsoft Windows 10 operating system, ensuring a realistic representation of contemporary malware. Automation was introduced to execute 2,916 malware samples in a virtual machine, with concurrent benign processes to maintain realism. The resulting 29,298 malicious memory dumps were balanced using the SMOTE algorithm [26]. The second stage involved transferring dump files to a Kali Linux machine for feature extraction using the Volatility Memory Analyzer (VolMemLyzer) [27], a Python project designed to facilitate the extraction of memory features, enhancing the analysis of malicious activities within a memory snapshot using the Volatility tool. VolMemLyzer efficiently captures the essential features for in-depth memory forensics and malware detection. Each sample is represented by a vector of numerical values derived from memory dump information, with 26 new extended feature groups, including Malfind (3), Ldrmodule (3), Handles (11), Process View (6), and Apihooks (3), totaling 58 features. Finally, the third stage encompassed feature extraction from memory dump files and creating a combined comma-separated values (CSV) file [1].

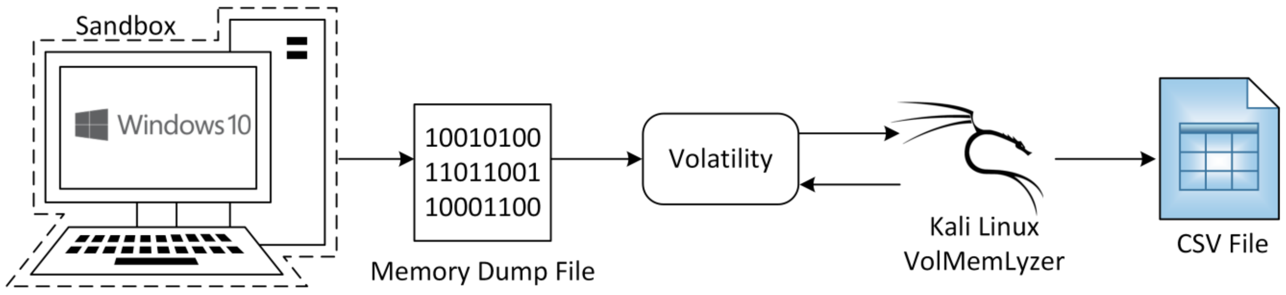

Figure 2. The CIC-MalMem-2022 Dataset Generation Processes

The CIC-MalMem-2022 dataset contains 58,596 memory dump samples. As shown in Table 1, the distribution of samples is well-balanced between benign and malware categories, with each class accounting for 50% of the total samples.

Table 1. The Number of Samples for CIC-MalMem-2022 Dataset

Class	**Number of Samples**	**Ratio %**
Benign	29,298	50
Malware	29,298	50
Total	58,596	100

The malware class is further divided into 3 categories and 15 subcategories, which are evenly distributed, as illustrated in Table 2.

Table 2. The Malware Subclasses in CIC-MalMem-2022 Dataset

Malware Details	Class	Number of Samples	Ratio %
Ransomware (9,791)	Ako	2,000	3.413
	Conti	1,988	3.413
	Maze	1,958	3.342
	Pysa	1,717	2.930
	Shade	2,128	3.632
Spyware (10,020)	180solutions	2,000	3.413
	CWS	2,000	3.413
	Gator	2,200	3.755
	TIBS	1,410	2.406
	Transponder	2,410	4.110
Trojan Horse (9,487)	Emotet	1,967	3.357
	Reconyc	1,570	2.679
	Refroso	2,000	3.413
	Scar	2,000	3.413
	Zeus	1,950	3.328

Ransomware (9,791 samples): This category includes strains such as Ako, Conti, Maze, Pysa, and Shade, which encrypt files and demand a ransom for decryption. Each strain has 1,717 and 2,128 samples, making up about 33.4% of the malware data.

Spyware (10,020 samples): This category consists of 180solutions, CWS, Gator, TIBS, and Transponder, which are programs that gather information about a person or organization without their knowledge. Each strain has between 1,410 and 2,410 samples, accounting for 34.3% of the malware data.

Trojan Horse (9,487 samples): This category includes Emotet, Reconyc, Refroso, Scar, and Zeus, programs that deceive users about their true intent, often appearing as legitimate software. Each strain has between 1,570 and 2,000 samples, making up 32.3% of the malware data.

3.4. Machine Learning Models

The ML models for classifying the CIC-Malmem-2022 dataset are generally explained in this section. We used RT, RF, J-48, NB, and XGBoost.

RT is an ML algorithm that constructs a decision tree by randomly selecting a subset of features at each node to split on. This randomness helps to reduce overfitting and improve generalization performance. RF enhances accuracy and reduces overfitting by averaging the predictions of multiple decision trees. C4.5, or J-48 in the Weka implementation, is a decision tree algorithm derived from the Iterative Dichotomiser 3 (ID3) algorithm. It selects crucial attributes from the information gain ratio and generates if-then rules from the trained trees. C4.5 is known for its robust performance across different datasets.

NB is a probabilistic classifier that applies Bayes' theorem for classification. It assumes that characteristics are conditionally independent given the class label, which simplifies the calculation of posterior probabilities. Despite its 'naive' assumption, NB often performs well in practice, especially for text classification tasks.

XGBoost is an optimized gradient-boosting algorithm widely used for classification and regression tasks. It builds a series of decision trees sequentially, where each tree attempts to correct the errors made by the previous ones. XGBoost is known for its computational efficiency and high performance on structured/tabular data.

While the dataset is not perfectly balanced, it is well-distributed and suitable for robust malware classification. Our study effectively utilizes this dataset to perform multilevel classification, yielding significant and reliable results.

4. Methodology

The system architecture and performance evaluations performed in this study are shown in Figure 3.

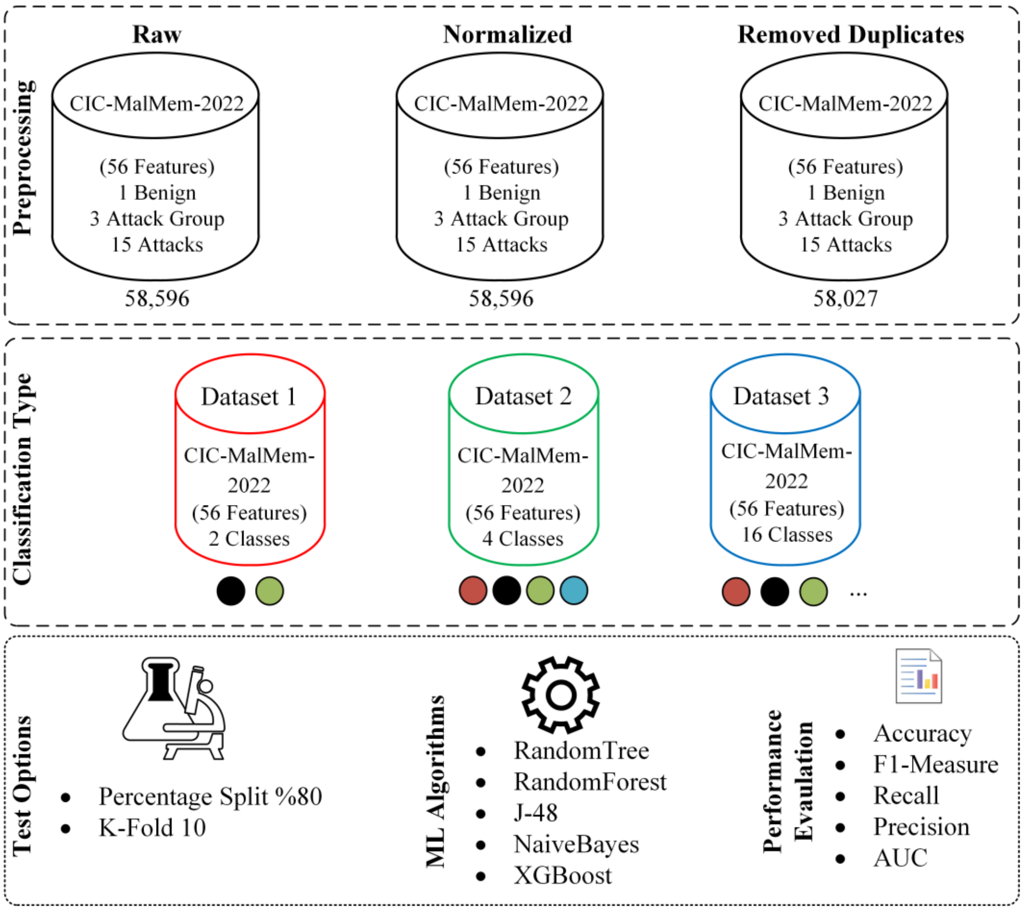

Figure 3. Proposed Malicious Software Detection System

Pre-processing steps, dataset creation, testing, and evaluation of machine learning (ML) methods were conducted using the CIC-MalMem-2022 dataset. Initially, all processes were applied to the raw dataset, and the corresponding tests were executed. It was noted that the initial low performance could be attributed to the lack of normalization and the presence of duplicate records. Consequently, all tests were repeated using a dataset from which duplicate records had been removed. The best performance was achieved with the duplicates removed dataset resulting in 58,027 records from the original 58,596 records.

The classification process began by organizing the data into binary classes (benign and malware). Subsequently, multiclass datasets with 4 and 16 classes were created. Various testing strategies were employed, including test split and k-fold cross-validation. For the percentage split test, 80% of the dataset was used for training, while the remaining 20% was used for testing. In the k-fold cross-validation method, the dataset was divided into 10 parts, with each part being used for testing in turn. The ML methods tested included Random Tree (RT), Random Forest (RF), J-48, Naive Bayes (NB), and XGBoost, all of which are commonly used in the literature. Performance metrics such as Accuracy, F1-Score, Recall, and Precision were calculated to assess the test results.

4.1. Data Preprocessing

Increasing the performance of machine learning methods depends significantly on data preprocessing steps. Therefore, analyzing the data well and carrying out the necessary pre-processing is very important. Normalization operations were performed on the dataset in our study. After this process, duplicate records in the data set were cleared. Thus, a more consistent and cleaner data set was obtained. At the end of these processes, sub-datasets such as binary, multiclass(4), and multiclass(16) were obtained from the data set containing 58,027 records, as shown in Figure 4.

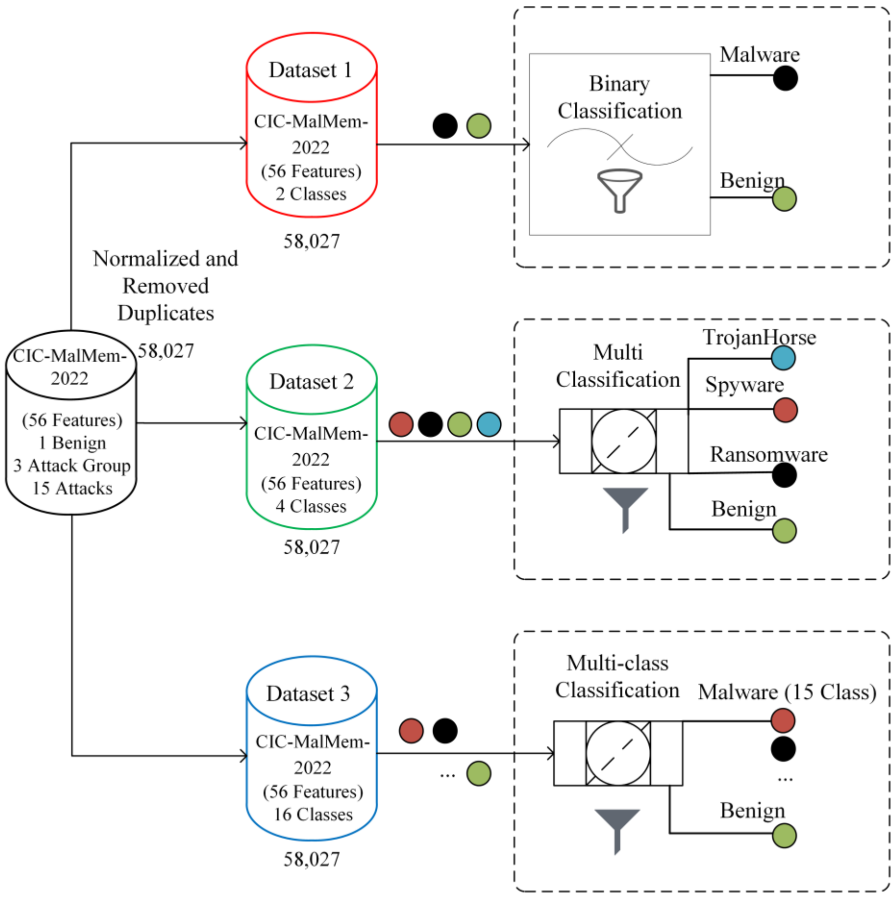

Figure 4. CIC-Malmem-2022 Dataset Preprocessing Operations

4.2. Experimental Setting

We utilized Weka version 3.9.6 [28] for training the machine learning models and conducting the evaluations. Table 3 details the hardware and software environment used in this study. The performance measurements were executed on a Windows operating system, powered by a CPU model i7-13700K @ 3.40GHz with 32 GB of RAM. The GPU employed was an NVIDIA GeForce® RTX 4070 Ti with 12 GB of memory.

Table 3. Hardware and Software Features for the Experimental Evaluations

Hardware / Software	**Features**
Operating System	Windows 11, 64 bit
Weka	3.9.6
CPU	13th Gen Intel(R) Core(TM) i7-13700K @ 3.40GHz
RAM	32 GB
Video Graphics Card	NVIDIA GeForce® RTX4070Ti

4.3. Evaluation Metrics

To assess the performance of the CIC-MalMem-2022 dataset with different algorithms, various performance metrics were used. The evaluations were performed using the WEKA [28]. Additionally, the confusion matrix used in computing these performance metrics is explained in Table 4.

Table 4. Confusion Matrix for Performance Calculation

Class/Attack Type		**Predicted Class**	
		Benign	**Malware**
True Class	**Benign**	True Positive (TP)	False Negative (FN)
	Malware	False Positive (FP)	True Negative (TN)

Accuracy: This metric indicates the percentage of correct predictions made by the model out of all predictions. It is calculated using the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN} \tag{1}$$

Recall: Recall measures the percentage of actual positives that were correctly identified by the model. It is calculated as:

$$\text{Recall} = \frac{TP}{(TP + FN)} \tag{2}$$

Precision: Precision indicates the percentage of positive predictions that were actually correct. It is computed as:

$$\text{Precision} = \frac{TP}{(TP + FP)} \tag{3}$$

F-Measure: The F1 score, representing the harmonic mean of Precision and Recall, is calculated as follows:

$$\text{F} - \text{Measure} = 2 \times \left(\frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}\right) \tag{4}$$

5. Experimental Results

This section presents the experimental studies conducted on the CIC-MalMem-2022 dataset. First of all, binary classification is given in the study. Then, the results obtained by considering multiclass classification with 4 and 16 classes are provided.

5.1. Binary Classification Results

For the binary classification task, algorithms like RF, J-48, and XGBoost demonstrate outstanding accuracy, with scores exceeding 99% in most cases. NB also performs reasonably well, albeit with slightly lower accuracy than other algorithms. XGBoost achieves perfect accuracy in both test modes, indicating its robustness in detecting obfuscated malware instances.

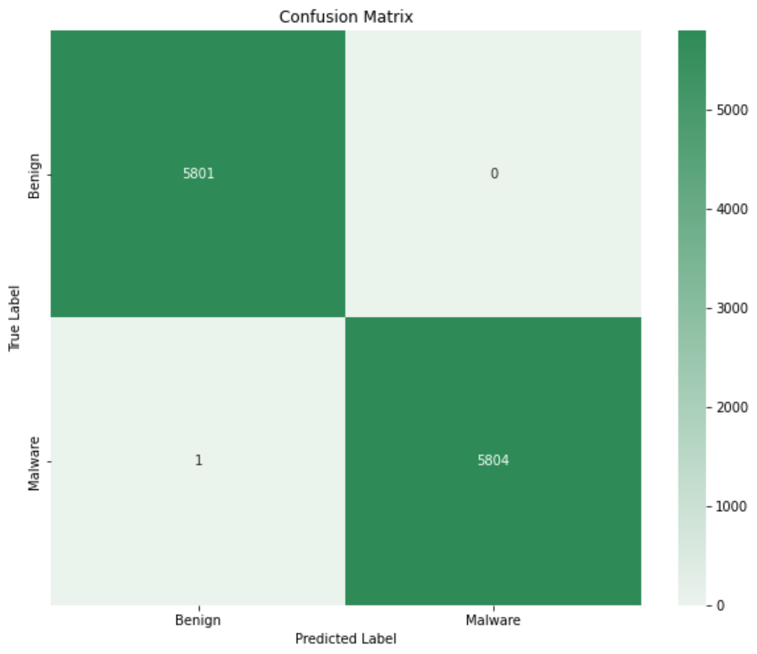

Figure 5. Confusion Matrix for XGBoost 80% Split for Binary Classification

Figure 5 shows the confusion matrix for binary classification utilizing XGBoost, demonstrating classification performance with 99.99% accuracy. The model showcases its exceptional ability to distinguish between different classes of obfuscated malware.

5.2. Multiclass Classification Results

In the multi-class classification scenarios (Multi-4 and Multi-16), RF, J-48, and XGBoost consistently deliver high accuracy rates above 70%. However, NB struggles to maintain comparable performance, especially in handling imbalanced datasets and distinguishing between multiple classes.

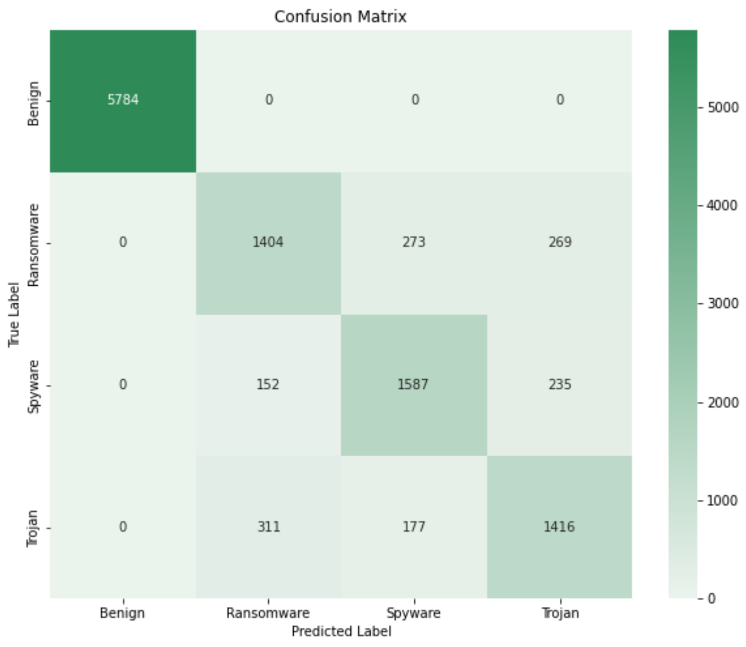

Figure 6. Confusion Matrix for XGBoost 80% Split for Multiclass(4) Classification

Figure 6 shows a confusion matrix with an accuracy of 87.79% and relatively balanced classification performance across all classes, as evidenced by the diagonal dominance and nonzero values outside the main diagonal. The classification report further reveals precision, recall, and F1-score metrics for each class, providing insights into the model's ability to identify instances of obfuscated malware within each category correctly.

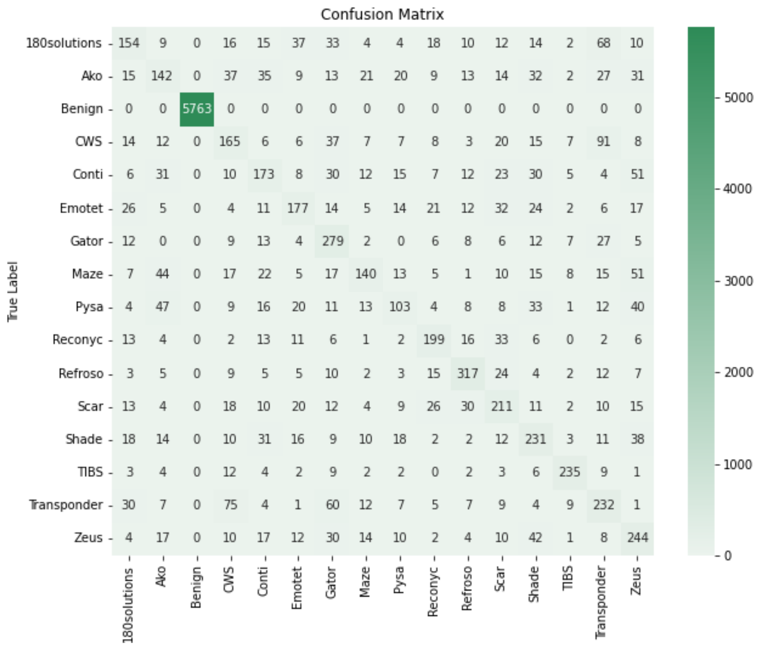

Figure 7. Confusion Matrix for XGBoost 80% Split for Multiclass(16) Classification

Figure 7 shows the confusion matrix for multi(16) classification with XGBoost. The XGBoost model trained with an 80% split achieved an accuracy of 75.49%. The confusion matrix indicates varying levels of precision and recall across different classes, with some classes showing more robust performance than others. Despite an overall macro-average F1-score of 0.54, indicating moderate classification performance, the model exhibits a relatively high mean absolute error of 1.5513, suggesting some discrepancy between predicted and actual values. Further optimization may be necessary to enhance the model's accuracy and robustness across all classes.

5.3. Performance Comparisons

Table 5 shows the performance results of ML algorithms applied to the CIC-MalMem-2022 dataset after removing duplicates. The results are presented for both an 80% / 20% train-test split and a 10-fold cross-validation setting, covering binary, multi(4), and multi(16) classification tasks. The evaluation metrics used include Accuracy (ACC), True Positive Rate (TPR), False Positive Rate (FPR), Precision, F-Measure.

Table 5. Experimental Results on the CIC-MalMem-2022 Dataset with Removing Duplicates

Dataset	Test Mode	Algorithms	ACC	TPR	FPR	Precision	F-Measure
CIC-MalMem-2022 (Binary-2)	percentage (80%)	Random Tree	99.96	1.00	0.000	1.00	1.00
		Random Forest	99.99	1.00	0.000	1.00	1.00
		J-48	99.97	1.00	0.000	1.00	1.00
		Naive Bayes	98.87	0.98	0.011	0.98	0.98
		XG-Boost	99.99	1.00	-	1.00	1.00
CIC-MalMem-2022 (Binary-2)	k-fold 10	Random Tree	99.96	1.00	0.000	1.00	1.00
		Random Forest	99.98	1.00	0.000	1.00	1.00
		J-48	99.96	1.00	0.000	1.00	1.00
		Naive Bayes	98.89	0.98	0.011	0.98	0.98
		XG-Boost	99.87	1.00	-	1.00	1.00
CIC-MalMem-2022 (Multi-4)	percentage (80%)	Random Tree	83.14	0.83	0.034	0.83	0.83
		Random Forest	87.61	0.87	0.025	0.87	0.87
		J-48	86.84	0.86	0.026	0.86	0.86
		Naive Bayes	67.80	0.67	0.064	0.71	0.63
		XG-Boost	87.79	0.88	-	0.88	0.88
CIC-MalMem-2022 (Multi-4)	k-fold 10	Random Tree	83.53	0.83	0.033	0.83	0.83
		Random Forest	87.48	0.87	0.025	0.87	0.87
		J-48	86.76	0.86	0.026	0.86	0.86
		Naive Bayes	68.24	0.68	0.063	0.72	0.64
		XG-Boost	82.84	0.83	-	0.83	0.83
CIC-MalMem-2022 (Multi-16)	percentage (80%)	Random Tree	70.12	0.70	0.011	0.70	0.70
		Random Forest	74.66	0.74	0.009	0.74	0.74
		J-48	74.76	0.74	0.009	0.74	0.74
		Naive Bayes	57.17	0.57	0.016	0.61	0.55
		XG-Boost	75.49	0.75	-	0.76	0.75
CIC-MalMem-2022 (Multi-16)	k-fold 10	Random Tree	70.16	0.70	0.011	0.70	0.70
		Random Forest	75.42	0.75	0.009	0.75	0.75
		J-48	75.12	0.75	0.009	0.00	0.75
		Naive Bayes	56.99	0.57	0.016	0.60	0.54
		XG-Boost	71.33	0.71	-	0.71	0.71

Overall, the results underscore the efficacy of ML algorithms, mainly RF, J-48, and XGBoost, in detecting obfuscated malware. These findings provide valuable insights for cybersecurity practitioners and researchers developing more effective defense mechanisms against sophisticated malware threats.

5.4. Discussion

Despite the successes reported in recent studies, several limitations are evident, indicating substantial opportunities for further research. Most approaches require large and balanced datasets, which are not always available, particularly for new or emerging malware variants. There are also concerns about the generalizability of these models across different platforms or under real-world conditions, as many were tested in controlled environments. Furthermore, the robustness of these systems against adversarial attacks, where attackers intentionally modify malware to evade detection, remains a significant challenge. These limitations underscore that while the field has made considerable advances, there is still much work to be done to enhance the efficacy, efficiency, and adaptability of ML models in the ever-evolving landscape of malware detection.

5.5. Comparative Analysis with Existing Methods

In the literature, some studies only perform binary classification with the CIC-MalMem-2022 dataset [1], [3], [4], [16], [23], [29]. For binary classification, it is seen that many studies achieve results above 99%. There are studies on binary and multi-class (4) classification in the literature [24]. Some studies deal with sub-malware types in the entire dataset, such as binary, multi-class (4) [24], and multi-class (16) [25], [18], [19], [30].

When we compare these studies with our proposed research, the best results were obtained with 87.79% accuracy with XGBoost in multi-class (4) classification and 75.49% accuracy in multi-class (16) classification, as shown in Table 6. These results demonstrate significant improvements over previous works, especially in the context of multilevel classification, where our model achieves higher accuracy rates in both multi-class (4) and multi-class (16) settings. This is a notable advancement given the complexity and challenges associated with multilevel malware classification.

Our study contributes to the literature by addressing the limitations of existing binary classification approaches and advancing the field through the implementation of multilevel classification. This approach allows for more granular detection of malware, providing a detailed and nuanced understanding of different malware types. It enhances the ability to identify and respond to new and emerging malware threats, which is crucial for developing robust and adaptable malware detection systems.

Table 6 summarizes the recent studies related to the CIC-MalMem-2022 dataset, highlighting the comparative performance of different models. Our results not only demonstrate the effectiveness of our approach but also emphasize the potential for further improvements in the field of malware detection.

Table 6. Recent Studies Related to the CIC-MalMem-2022 Dataset

Authors	**Model Tested**	**Best Model**	**Accuracy (%)**		
			Binary	**Multiclass (4)**	**Multiclass (16)**
Carrier et al. [1]	NB, RF, DT and meta-learnerLR	LR	99.00	-	-
Yihan et al. [3]	Hierarchical Detection Model	DNN	99.67	-	-
Ghazi and Raghava [4]	RF, SVM, kNN	RF	99.99	-	-
Dener et al. [16]	RF, DT, GBT, LR, NaiveBayes,	LR	99.97	-	-
Smith et al. [23]	DT, RF, Ada Boost, KNN, SGD,	DT	99.99	-	-
Roshan and Zafar. [29]	NN, Adaptive RF, KNN, Voting	EnsAdp_CIDS	99.85	-	-
Mezina and Burget [24]	LR, DT, MLP, kNN, SVM, D-CNN	D-CNN	99.99	83.53	-
Cevallos-Salaset al. [18]	DT, RF, SVM, LDA, GNB	LR+DNN	99.70	75.40	68.20
Shafin et al. [19]	RobustCBL, CompactCBL	RobustCBL	99.96	84.56	72.60
Abualhaj et al. [30]	KNN	KNN	99.97	82.21	66.93
Roy et al. [25]	Extra Trees, XGBoost, SVM, kNN,	MalHyStack	99.98	85.04	70.29
Our Model 2024	RT, RF, J-48, NaiveBayes, XGBoost	XGBoost	**99.99**	**87.79**	**75.49**

6. Conclusion

In this study, we comprehensively evaluated ML algorithms for detecting obfuscated malware using the CIC-MalMem-2022 dataset. Our analysis encompassed binary and multi-class classification tasks under different experimental conditions, including percentage splits and 10-fold cross-validation. The results highlight the effectiveness of RF, J-48 (C4.5), and XGBoost algorithms in achieving high accuracy across various classification tasks. These algorithms consistently outperformed others, showcasing their robustness in identifying obfuscated malware patterns. Random Tree also exhibited commendable performance, albeit slightly lower than the algorithms above. Furthermore, Naive Bayes demonstrated competitive performance but faced challenges handling multi-class classification and imbalanced datasets, as evidenced by lower accuracy rates and higher mean absolute error values.

Overall, the findings show the importance of advanced ML techniques for enhancing obfuscated malware detection capabilities. The identified top-performing algorithms can be useful tools for cybersecurity researchers to develop more effective defense mechanisms against obfuscated malware. Future research efforts could focus on fine-tuning model hyperparameters, optimizing data preprocessing techniques, and further exploring ensemble learning approaches to improve detection accuracy and resilience against obfuscated malware attacks.

References

[1] T. Carrier, P. Victor, A. Tekeoglu, and A. Habibi Lashkari, "Detecting Obfuscated Malware using Memory Feature Engineering," in *International Conference on Information Systems Security and Privacy*, 2022. doi: 10.5220/0010908200003120.

[2] Z. A. El Houda, "Cyber Threat Actors Review: Examining the Tactics and Motivations of Adversaries in the Cyber Landscape," in *Cyber Security for Next-Generation Computing Technologies*, 2024. doi: 10.1201/9781003404361-5.

[3] Y. Li, Z. Liu, X. Guan, Z. Wang, X. Guo, and S. Wang, "Hierarchical Obfuscation Malware Detection Method Based on Deep Learning," in *EEI 2022 - 4th International Conference on Electronic Engineering and Informatics*, 2022.

[4] M. R. Ghazi and N. S. Raghava, "Machine Learning Based Obfuscated Malware Detection in the Cloud Environment with Nature-Inspired Feature Selection," in *2022 5th International Conference on Multimedia, Signal Processing and Communication Technologies, IMPACT 2022*, 2022. doi: 10.1109/IMPACT55510.2022.10029271.

[5] M. A. Hossain and M. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity," *Cybersecurity*, vol. 7, no. 1, 2024, doi: 10.1186/s42400-024-00205-z.

[6] B. Janet, A. Nikam, and J. A. Kumar R, "Real Time Malicious URL Detection on twitch using Machine Learning," in *Proceedings of the International Conference on Electronics and Renewable Systems, ICEARS 2022*, 2022. doi: 10.1109/ICEARS53579.2022.9751862.

[7] M. Hakimi, E. Ahmady, A. K. Shahidzay, A. W. Fazil, M. M. Quchi, and R. Akbari, "Securing Cyberspace: Exploring the Efficacy of SVM (Poly, Sigmoid) and ANN in Malware Analysis," *Cognizance Journal of Multidisciplinary Studies*, vol. 3, no. 12, 2023, doi: 10.47760/cognizance.2023.v03i12.017.

[8] S. Altaha and K. Riad, "Machine Learning in Malware Analysis: Current Trends and Future Directions," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 1, 2024, doi: 10.14569/IJACSA.2024.01501124.

[9] V. Vijayaraj, M. Balamurugan, and M. Oberai, "Machine learning approaches to identify the data types in big data environment: An overview," *The Scientific Temper*, vol. 14, no. 03, 2023, doi: 10.58414/scientifictemper.2023.14.3.60.

[10] M. Azeem, D. Khan, S. Iftikhar, S. Bawazeer, and M. Alzahrani, "Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches," *Heliyon*, vol. 10, no. 1, 2024, doi: 10.1016/j.heliyon.2023.e23574.

[11] A. Nugraha and J. Zeniarja, "Malware Detection Using Decision Tree Algorithm Based on Memory Features Engineering," *Journal of Applied Intelligent System*, vol. 7, no. 3, 2022, doi: 10.33633/jais.v7i3.6735.

[12] Akoh Atadoga, Enoch Oluwademilade Sodiya, Uchenna Joseph Umoga, and Olukunle Oladipupo Amoo, "A comprehensive review of machine learning's role in enhancing network security and threat detection," *World Journal of Advanced Research and Reviews*, vol. 21, no. 2, 2024, doi: 10.30574/wjarr.2024.21.2.0501.

[13] L. Zhang and Q. Yan, "Detect malicious websites by building a neural network to capture global and local features of websites," *Comput Secur*, vol. 137, 2024, doi: 10.1016/j.cose.2023.103641.

[14] M. Kozák, M. Jureček, M. Stamp, and F. Di Troia, "Creating valid adversarial examples of malware," *Journal of Computer Virology and Hacking Techniques*, 2024, doi: 10.1007/s11416-024-00516-2.

[15] K. Shaukat, S. Luo, and V. Varadharajan, "A novel machine learning approach for detecting first-time-appeared malware," *Eng Appl Artif Intell*, vol. 131, 2024, doi: 10.1016/j.engappai.2023.107801.

[16] M. Dener, G. Ok, and A. Orman, "Malware Detection Using Memory Analysis Data in Big Data Environment," *Applied Sciences (Switzerland)*, vol. 12, no. 17, 2022, doi: 10.3390/app12178604.

[17] M. A. Hossain *et al.*, "AI-enabled approach for enhancing obfuscated malware detection: a hybrid ensemble learning with combined feature selection techniques," *International Journal of System Assurance Engineering and Management*, 2024, doi: 10.1007/s13198-024-02294-y.

[18] D. Cevallos-Salas, F. Grijalva, J. Estrada-Jimenez, D. Benitez, and R. Andrade, "Obfuscated Privacy Malware Classifiers Based on Memory Dumping Analysis," *IEEE Access*, vol. 12, 2024, doi: 10.1109/ACCESS.2024.3358840.

[19] S. S. Shafin, G. Karmakar, and I. Mareels, "Obfuscated Memory Malware Detection in Resource-Constrained IoT Devices for Smart City Applications," *Sensors*, vol. 23, no. 11, 2023, doi: 10.3390/s23115348.

[20] L. N. Tidjon and F. Khomh, "Reliable malware analysis and detection using topology data analysis," *arXiv preprint arXiv:2211.01535*, 2022.

[21] H. Naeem, S. Dong, O. J. Falana, and F. Ullah, "Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification," *Expert Syst Appl*, vol. 223, 2023, doi: 10.1016/j.eswa.2023.119952.

[22] M. Al-Qudah, Z. Ashi, M. Alnabhan, and Q. Abu Al-Haija, "Effective One-Class Classifier Model for Memory Dump

Malware Detection,” *Journal of Sensor and Actuator Networks*, vol. 12, no. 1, 2023, doi: 10.3390/jsan12010005.

[23] D. Smith, S. Khorsandroo, and K. Roy, “Supervised and Unsupervised Learning Techniques Utilizing Malware Datasets,” in *2023 IEEE 2nd International Conference on AI in Cybersecurity, ICAIC 2023*, 2023. doi: 10.1109/ICAIC57335.2023.10044169.

[24] A. Mezina and R. Burget, “Obfuscated malware detection using dilated convolutional network,” in *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, 2022. doi: 10.1109/ICUMT57764.2022.9943443.

[25] K. S. Roy, T. Ahmed, P. B. Udas, M. E. Karim, and S. Majumdar, “MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis,” *Intelligent Systems with Applications*, vol. 20, 2023, doi: 10.1016/j.iswa.2023.200283.

[26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, 2002, doi: 10.1613/jair.953.

[27] A. H. Lashkari, B. Li, T. L. Carrier, and G. Kaur, “VolMemLyzer: Volatile Memory Analyzer for Malware Classification using Feature Engineering,” in *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge, RDAAPS 2021*, 2021. doi: 10.1109/RDAAPS48126.2021.9452028.

[28] E. Frank, M. A. Hall, and I. H. Witten, “The WEKA Workbench Data Mining: Practical Machine Learning Tools and Techniques,” *Morgan Kaufmann, Fourth Edition*, 2016.

[29] K. Roshan and A. Zafar, “Ensemble adaptive online machine learning in data stream: a case study in cyber intrusion detection system,” *International Journal of Information Technology (Singapore)*, 2024, doi: 10.1007/s41870-024-01727-y.

[30] M. M. Abualhaj, A. A. Abu-Shareha, Q. Y. Shambour, A. Alsaaidah, S. N. Al-Khatib, and M. Anbar, “Customized K-nearest neighbors’ algorithm for malware detection,” *International Journal of Data and Network Science*, vol. 8, no. 1, 2024, doi: 10.5267/j.ijdns.2023.9.012.

Conflicts of Interest

Authors declare no conflict of interest.

Ethical Approval

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of Data and Material

The CIC-MalMem-2022 dataset is accessible at: https://www.unb.ca/cic/datasets/malmem-2022.html

Plagiarism Statement

This article has been scanned by iThenticate ™.