"Unmasking Memory Malware: A Comparative Analysis of Individual Machine Learning and Deep Learning Using Ensemble Approaches"

MSc Research Project
Programme Name

Samita Ramesh Babu
Student ID: 22132201

School of Computing
National College of Ireland

Supervisor: Evgeniia Jayasekera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	SAMITA RAMESH BABU		
Student ID:	22132201		
Programme:	MSc Cybersecurity	**Year:**	2023-2024
Module:	MSc Research Project		
Supervisor:	Evgeniia Jayasekera		
Submission Due Date:	31 Jan 2024		
Project Title:	"Unmasking Memory Malware: A Comparative Analysis of Individual Machine Learning and Deep Learning using Ensemble Approaches"		
Word Count:	6106	**Page Count:**	20

Signature: Samita

Date: 31 Jan 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

"Unmasking Memory Malware: A Comparative Analysis of Individual Machine Learning and Deep Learning Using Ensemble Approaches"

Samita Ramesh Babu
22132201

Abstract

Abstract: Obfuscated Malware is malware that hides to avoid detection. Cyber-attacks are constantly prevailing in recent years even sometimes it is undetected by anti-virus software. The study involves detecting memory malware using machine and deep learning models using ensemble methods. The implementation is done by preprocessing and sampling of data of memory malware detection optimizing the representation of memory samples for effective analysis. Experiments were performed on a variety of machine learning algorithms and deep learning method, such as MLP Classifier, Adaboost, Gaussian Naive Bayes, Bagging classifier, SGD both individually and combined. Our findings reveal that ensemble methods performed compared to other models used in this research. Bagging classifier is outperformed individual algorithms by showing 92% accuracy. Then in combination of models Bagging and GNB showed 89% accuracy. The performance of these algorithms is evaluated based on metrics such as accuracy, precision, recall, and F1 score. This research can be guide for cybersecurity professionals seeking to implement efficient memory malware detection strategies.

Key Words: Obfuscate Malware Memory Malware Analysis, Cybersecurity, Ensemble Learning, Machine Learning

1 Introduction

Malware has been created to attack, damage, or disable mobile phones, computers, apps, or systems using a code or script. Memory-based attacks can execute in real time, enabling hackers to insert malicious code. It exploits zero-day vulnerabilities. Nowadays, Obfuscation technique makes it difficult to read. There were 470.01 million malwares were detected and in 2022 around 30 million malwares were detected [2]. Memory analysis data can yield valuable insights into the behavior and patterns of malicious software. This is a result of the different traces malware leaves on memory. Because of this, one of the topics that needs to be researched in malware detection is the memory analysis method.

Encryption: Obfuscators can encrypt the malware code to make it difficult for static analysis tools to understand. Randomization: Obfuscators can randomize the layout of the malware code in memory to make it difficult for dynamic analysis tools to trace its execution.

Code packing: Obfuscators can pack the malware code into a more compact form, which can make it more difficult for antivirus scanners to detect.
Control Flow
Jumping: Obfuscators can insert unnecessary jumps into the malware code to make it more difficult for dynamic analysis tools to follow the execution flow.
Splicing: Obfuscators can splice together different pieces of code to make it more difficult for static analysis tools to understand the overall logic of the malware. Code injection: Obfuscators can inject code into the malware at runtime to bypass security checks.

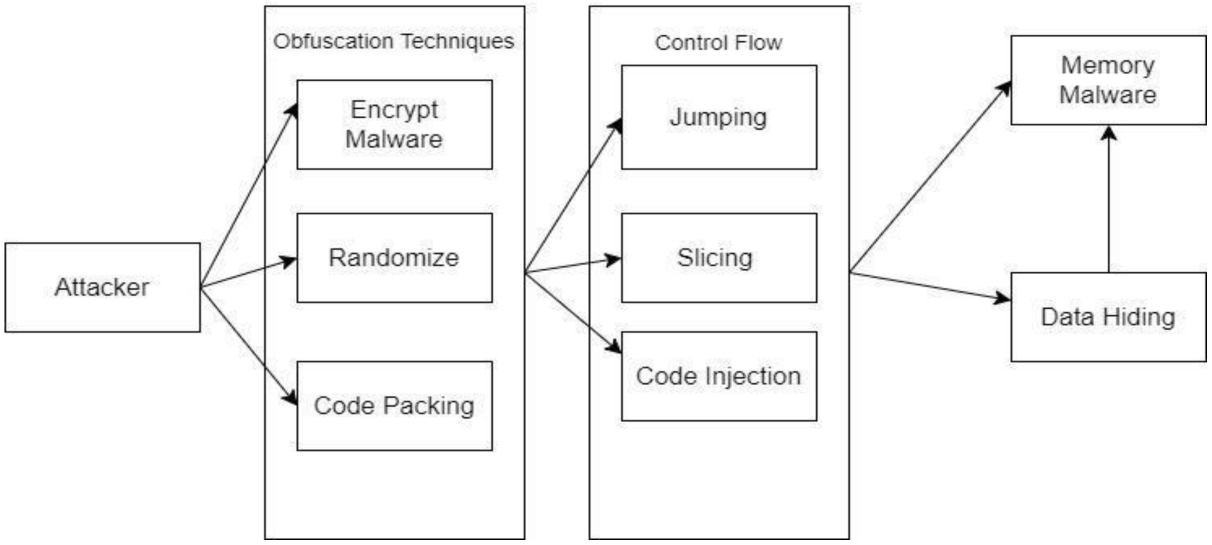

Figure 1: Overview of Possibilities of Obfuscation Memory Malware Techniques

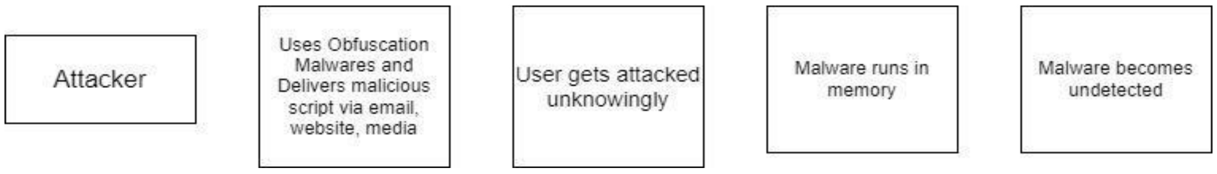
Attacker
Uses Obfuscation Malwares and Delivers malicious script via email, website, media
User gets attacked unknowingly
Malware runs in memory
Malware becomes undetected

Figure 2: Stages of an Obfuscation Memory Malware Attack

Research Questions

What is the impact of combining diverse machine learning models and deep learning models, such as MLP, Adaboost, Gaussian Naive Bayes, Bagging, and Stochastic Gradient Boosting, in a hybrid approach for the detection of memory malware, and how does this approach compare to individual models and existing methods in terms of accuracy and efficiency?

In this research, we aim to detect memory malware analysis using MLP Classifier, Adaboost, Gaussian Naive Bayes, Bagging classifier, SGD were the five different machine learning and deep learning techniques. Experimenting with individual machine learning and deep learning algorithms to understand their standalone performance. Determining the most efficient algorithm or combination of algorithms of detecting memory malware involves factors such as accuracy, speed and other evaluation metrics. Additionally, the research seeks to identify the most accurate algorithm and combination of algorithms for memory malware detection. Additionally, evaluating the trade-off between accuracy and computational efficiency, especially within a limited time frame. Contributing to the advancement of memory malware detection methodologies, considering the evolving threat landscape. MLP, Gaussian Naive Bayes, and SGD Classifier are individual machine learning algorithms. They might be simpler and faster, but their performance can depend on the data and the chosen hyperparameters. Ensemble methods combine multiple base learners to form a stronger model. They often perform well in practice due to their ability to handle different aspects of the data. Bagging can reduce variance and improve stability.

The rest of paper is organized as follows. Literature Review are given section 2, a brief review of Machine and Deep Learning methods and its implications in the memory malware detection. The survey of all related works is conducted. In section 3, detailed explanations of machine and deep learning methods used to include preprocessing of data, feature extraction. In section 4, flowchart of the malware detection, details of framework used. In section 5, includes the details of Dataset used, training and testing using different algorithms are given. Evaluation metrics and experiment results are given in section 6.

2 Related Work

In this section a detailed study of ensemble approaches, machine learning and deep learning approaches from the existing studies is searched using Google Scholar Database. The articles found were scrutinized based on memory malware using ensemble methods, machine learning and deep learning,

2.1 Machine Learning Approaches

Vashishtha et al. proposed a model using different voting process. In this research they converted image files to analyzed. The experiment was based on voting ensembling techniques.[13] A hybrid model MalHyStack was incorporated with ensemble learning was proposed by Roy et al. The model detected obfuscated malware. Pearson correlation analysis has improved accuracy of model a bit more along with reducing the computational time. This model can still be made more efficient and reliable in future according to the authors [10].

Naeem et al. proposed a deep stacked ensemble by combining with CNN and MLP as output and input. This model was evaluated using three datasets Dumpware10 dataset, CIC-

MalMem2022 and real-world dataset. Windows malware memory dumps [17]. The malware detection model based on ensemble learning was performed. The model was trained using minimum features extracted from file. These methods performed well than classification models but could have used more techniques [18]. In this study, many algorithms were experimented SVM, KNN and Random Forest for memory malware analysis, the accuracy of these models were around 98.5% but the false were also high [19]. Khalid et al. proposes a detection of fileless malware by analyzing features for main memory. The research uses feature analysis using machine learning. Random Forest Decision Tree, Support Vector Machine, Logistic Regression, XGBoost, and Gradient Boosting were used to experiment. VirusShare, AnyRun, PolySwarm, HatchingTriage, and JoESandbox were the five datasets used. Authors suggest more techniques can be incorporated for further studies [14]. Bruna conducted a comparative study using two datasets CIC-Evasive-PDFMal2022 and CIC-MalMem-2022. Random Forest achieved low computational time In the other dataset, was bigger and computational time was high and yielded around 74% detection rate.SVM classification and perform using other machine learning to yield better results [9]. A comprehensive analysis was made by Saad et al. in detecting adversarial malware using machine learning and also stating that behavioral analysis can detect malware in future [20].

2.2 Deep Learning Approaches

Gombe et al. proposed a model using cRGB_Mem which trains RGB images that is been generated in memory allocation patterns in CNN. This RGB-CNN model is distinguished between benign and malware. This model predicts Android malware detection between (R01) known and (R02) unknown features but the detection accuracies for R01 and R02 are not perfect. The memory allocation pattern is based in allocation address which may be inaccurate [1]. The paper proposed binary classification using CIC-MalMem-2022 dataset to detect malware. This study also provides classification using machine learning and deep learning in memory analysis using big data approach [2].
Another study proposed memory-based method to detect malwares that reside in the computer’s memory. CLAHE and wavelet transform were two techniques for feature extraction. This model displayed accuracy and precision using less training time. Further, this method can be improved in terms of accuracy by using different feature selection methods and computation cost [3]. Another paper proposed machine and deep learning approaches to detect Android attacks. Using CICAndMal2017 dataset and LSTM achieved 99.4% accuracy using Drebin dataset [4].
Bozkir et. al proposed a method that focused on memory analysis by capturing memory dumps. The model uses GIST and HOG as image descriptors and used. The UMAP based manifold learning strategy has made the model even better by improving accuracy by detecting unknown malware. The results tend to show that transforming memory dumps to initial images yielded 4096px [5]. Basirah and Sana proposed a rootkit detection model using memory analysis Using KNN and LSTM algorithm. KNN performed well in less execution time. Deep Learning models proved to provide 18% more accuracy which is moderate. There

were few limitations as the dataset was too small due to which it created overhead and causes low accuracy [6].
Xu et al. introduced a hardware assisted malware detection framework using two types of malwares kernel rootkits and memory corruption attacks. They have used function call and entire program epoch for the detection. By changing the histogram bin size, the entire program epoch changes but the function call remains resilient. This needs trial and error check each time to check which limited the automation. Random forest and Logistic Regression was performed. Although this framework provides 99% detection rate for memory corruption attacks and 100% detection for kernel rootkits. The methods seem to overfit the framework [7].
Klaib et al. used memory dump malware using supervised machine learning algorithms such as KNN. This paper used two scenarios with CIC-MalMem-2022 dataset and correlation matrix to compare which one was effective [8]. Bruna conducted a comparative study using two datasets CIC-Evasive-PDFMal2022 and CIC-MalMem-2022. In first dataset, KNN provides better results in hamming distance. Also tried using oversampling and undersampling but was not able to improve the results. The drawback author was not able to balance the dataset [9].

Vinayakumar et al. proposed MLAs and deep learning architecture based on Static Analysis, Dynamic Analysis, and image processing techniques for malware detection and ScaleMalNet for detecting and categorize zero-day malwares. It relies on domain knowledge features for dynamic analysis, lacks robustness against adversarial attacks, and encounters challenges in handling imbalanced malware datasets. Moreover, its image processing approach may need enhancement for flexibility, especially concerning varying image sizes [11].
A study proposed MDCD that is dynamic malware detection solution for cloud environments employs a lightweight agent to collect runtime utilization and utilizes memory forensics for memory object information. The multi-CNN model achieves remarkable average accuracy, precision, recall, and F1 Score. The method outperforms existing solutions, effectively detecting multiple malicious processes with minimal deployment effort. Although, computational cost impacted the performance of VM. Increase of VM affects scalability of dynamic malware [12].

Signature and Machine Learning approach was used to analyze the execution binary. Li et al. used dynamic analysis and deep learning approach. Then later CNN, was use to detect the memory snapshots from the virtual machine [15].

Models Used	Citation
Voting Ensemble	[13][17]
MalHyStack	[10]
KNN	[6][8]
Decision Tree	[6][8]
CNN	[15][6][1]
Feature Analysis	[14]

Table 1: Models Used in existing study

The literature review reveals significant insights into memory malware detection methodologies. Ensemble approaches, such as the hybrid model combining MLP, Adaboost, and Gaussian Naive Bayes, demonstrated promising results. Various studies explored machine learning and deep learning techniques, including CNN, SVM, and dynamic analysis, emphasizing the need for effective models. In the existing study, balancing the dataset was not easy and algorithms shows accuracy, but false positives are bit high. Notable works addressed obfuscation techniques, encryption, and code packing, highlighting the evolving threat landscape and the importance of accurate, efficient detection methods in countering the rising tide of malware.

3 Research Methodology

In this section, a detailed explanation of project is explained. Many machine learning models are used for comparative analysis of which algorithm performs the best out of others. MLP, AdaBoost, GNB, Bagging and SGB are experimented individually and in combinations to identify memory malware. To build the models, it comprises of 4 parts mainly, Data Pre-processing, Sampling of Data, Malware Families and Classification. Although, a overview of all the types and subtypes of each malware is mentioned but will be focusing on more three particular malwares such as Trojan Horse, Spyware and Ransomware.

3.1 Data Preprocessing:

The initial phase of research conducted to make the CIC-MalMem2022 dataset suitable for classification. This Is done to improve the efficiency of classification models. The CIC MalMem2022 dataset is balanced dataset which consists of two classes benign and malware. Next step is categorized benign and malwares into four categories such as Benign, Ransomware, Spy and Trojan. The Label Encoder process is used for converting categorical class values to numerical values. Benign and Malware are assigned as 0 and 1 in class column. By creating an output column for the Category.

3.2 Sampling of Data

The dataset is each category is given a unique value such as 0,1,2 and 3. Each correspond to benign, ransomware, spyware and trojan. The dataset used in this research paper has more benign values than each malware. Therefore, it indicates a class imbalance in the dataset. This imbalance can potentially affect the performance of machine learning models, particularly in their ability to accurately predict minority classes (malware instances) during training and validation. To address this imbalance, Synthetic Minority Over-sampling Technique (SMOTE) is applied to generate synthetic samples of Trojan, Spyware and Ransomware categories. This is done balance the class distribution. Thus, making the model more robust and unbiased. In previous study **[9]**, were not able to perform sampling of data. In this, we were able to perform oversampling using SMOTE technique.

Further, the dataset is split into training, validation, and testing. The first split is of 60% of data is used for training and 40% for testing. Then second split is 50% for training and 50% for testing.

3.3 Malware Families

A Trojan Horse refers to a kind of software that operates discreetly in the background pretending to be legitimate. When a user downloads a file, it replicates itself without authorization, across all directories where the user has access. There are types of Trojan Horses that have emerged over time.

Another type is Emonet, which emerged in 2014 as banking malware designed to gather information by sniffing network traffic. As time passed it transformed into a platform of facilitating the installation of malware. Emonet can also. Manage botnets while possessing some worm characteristics.

Refroso is another type of trojan horse that significantly disrupts Windows systems. It handles access connections. Performs distributed denial of service (DDoS) attacks. It even automatically modifies firewall settings by deleting registry entries.
There are activities that it conceals. It also redirects web browsers to harmful websites.

Scar: It functions, as a Trojan horse enabling the installation of types of malwares on the device. It downloads a list of URLs that contain files with the ".exe" extension allowing for the download of malware. Additionally, it can carry out operations like gathering information from the device and altering system settings.

Reconyc As a Trojan horse its primary function is to download forms of malware onto the compromised device. Like malware it spreads through websites or as attachments to other files. It can also restrict access to tools within the operating system such as Command Prompt, Task Manager and Registry Editor.

Spyware: This category encompasses malware designed to record user information and activities. Then transmit them to third parties. Typically, spyware collects data about a user’s browsing patterns and online activities. Within our dataset we have identified five types of spyware.

180Solutions; Also known as Zango this spyware monitors internet activities including user movements visited URLs and cookies. It utilizes this collected information, for displaying pop up ads and targeted advertisements.

CoolWebSearch (CWS); This browser hijacker first emerged in 2003.
There is a type of software called CoolWebSearch that transfers data collected from web browsers to networks. It comes in versions, such, as DataNoter, BootConf, PnP, Winres,

SvcHost and MSInfo. Each version has its functions like monitoring websites access ensuring CoolWebSearch doesn't appear on whitelists and downloading adware. Then there's Gator which's an adware also known as Gain AdServer. It can pretend to be a virus to replicate itself. Can also download spyware programs and perform updates. Like adware it tracks user activities and displays targeted ads and pop ups. Gator can use up a lot of hard disk space causing memory wear.

Another one is Transponder which is spyware that installs itself as a Browser Helper Object (BHO) distributed with third party software. During its setup it collects information about the device and user ID. After that it monitors activities such, as user movements visited URLs cookies usage etc.. Sends them to the server. It also creates pop up banners.

Lastly TIBS is a malware referred to as TIBS dialer. It spreads through email attachments and unreliable websites. Its purpose is to make paid calls to adult websites using the modem.
There is a background process running on the device that doesn't impact its performance. It shows up through situations, like connections, unwanted downloads, and hidden internet connections.

Ransomware is a type of software that aims to extort money from users. It restricts user access by encrypting files, disks, or other data on the device. To regain access users are required to pay a fee to remove the encryption. However, there's no guarantee that paying the specified ransom will always result in accessing the encrypted data. Ransomware is currently a growing issue [2].

These are five types of spyware included in the dataset;
1. Conti; This ransomware emerged in 2020. Infiltrates networked drives through phishing emails. When clicked it downloads Bazar backdoor and IcedID Trojan horse onto targeted machines. It encrypts SMB type files using AES 256 with up to 32 threads during encryption process. It ignores files with dll, exe, lnk and sys extensions while encrypting and also deletes shadow copies of encrypted files while preventing their restoration.
2. Maze; First observed in 2019 Maze is typically distributed via phishing emails containing macros attached as files or through vulnerabilities found in networks such, as RDP servers and Citrix/VPN servers.
It is also available, in the form of a PE binary (dll, exe). It uses ChaCha20 stream ciphers and RSA 2048 public encryption keys to encrypt files. That's why it is sometimes referred to as ChaCha ransomware. The individuals behind Maze ransomware publish encrypted documents on their websites.
Pysa; This particular type of ransomware emerged in 2018. Cannot spread on its own. It is also known as Mespinoza. By employing Brute Force attacks against RDP servers and Active Directory phishing emails manage to infiltrate machines. Pysa utilizes an encryption method that combines AES CBC and RSA algorithms. Encrypted files are stored with the Pysa extension. Additionally it removes shadow copies of encrypted files to prevent their restoration.

Ako; This ransomware appeared in 2020. Gains access to machines through phishing emails. It is also known as MedusaReborn. Ako is distributed via an encrypted zip file in the folders src file. MD5, SHA 1 and SHA 256 are employed for encrypting file types excluding exe, dll, sys, ini, lnk, key and rdp files. Upon infiltration Ako drops a text containing the ransom note alongside a folder named "id.key" that holds the encryption key, on the target desktop.
Shade is a type of ransomware that emerged in 2019 and gains access, to computers through phishing emails. It is also referred to as Troldesh. Shade spreads through a zip file written in Javascript. Employs two keys generated using AES 256 in CBC mode to encrypt both the content and filenames of individual files. Additionally, it is notorious for leaving notes with extensions, on infected computers.

3.4 Methods Used

MLP is a type of artificial neural network that can learn complex nonlinear relationships between features and targets. AdaBoost is an ensemble learning algorithm that combines multiple weak learners to create a strong learner. GNB is a probabilistic classifier that assumes that the features of a data point are independent of each other. SGD is an algorithm for training linear models. Bagging is an ensemble learning algorithm that creates multiple copies of a base learner and trains each copy on a different subset of the training data. All the models are trained separately, and different parameters were used for all algorithms. Another part of methodology includes combination of these models, with different parameters to see which hybrid performs best in identifying memory malware.

4 Design Specification

4.1 Flow Chart

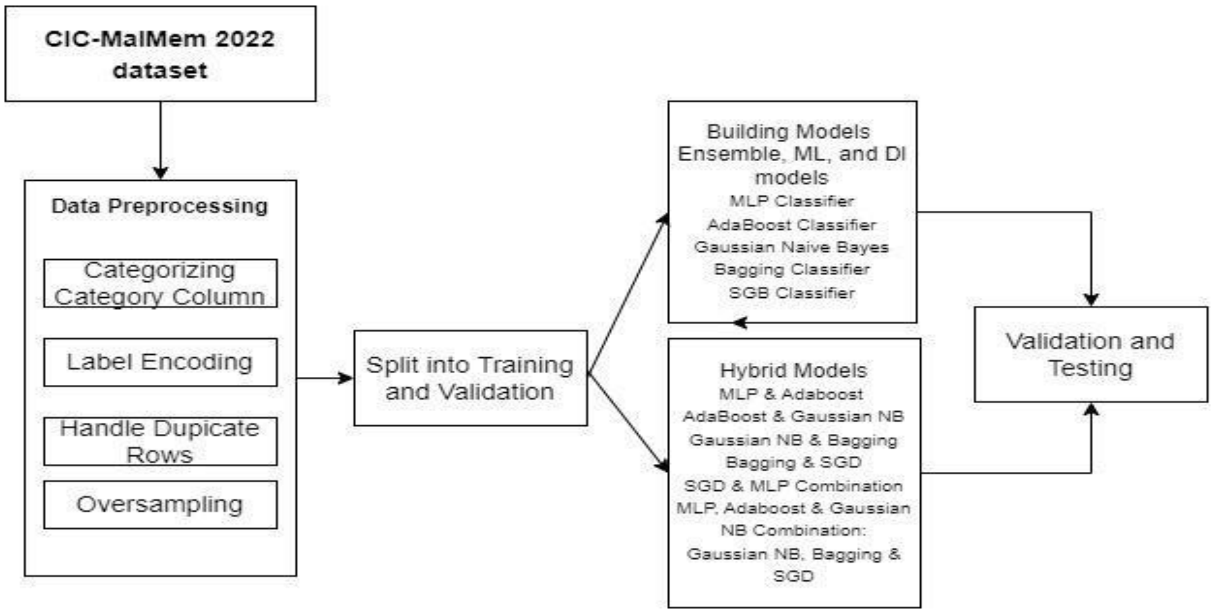

Figure 3: Flow Chart

4.2 Chart of Benign and different Malwares

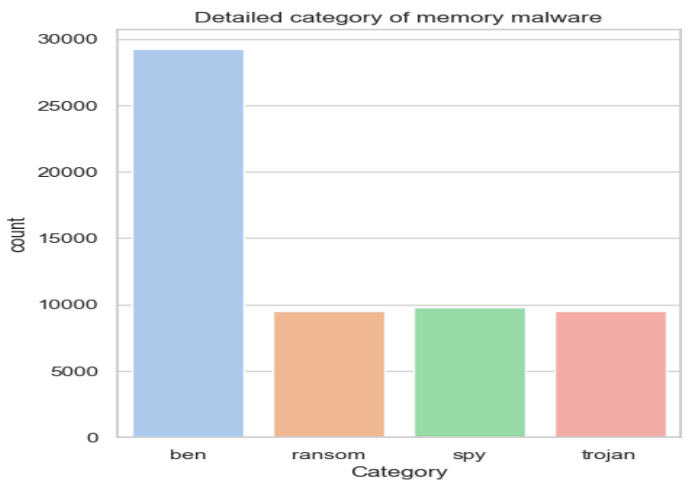

Figure 3: Detailed Graph of Categories and Malwares

Based on the dataset, consists of 4 categories such as benign (ben), ransomware (ransom), spyware(spy) and trojan horse (trojan). Based on the graph, benign is the majority category and rest of the malwares are almost in the equal ratio.

4.3 Comparison of Category with respect to Class

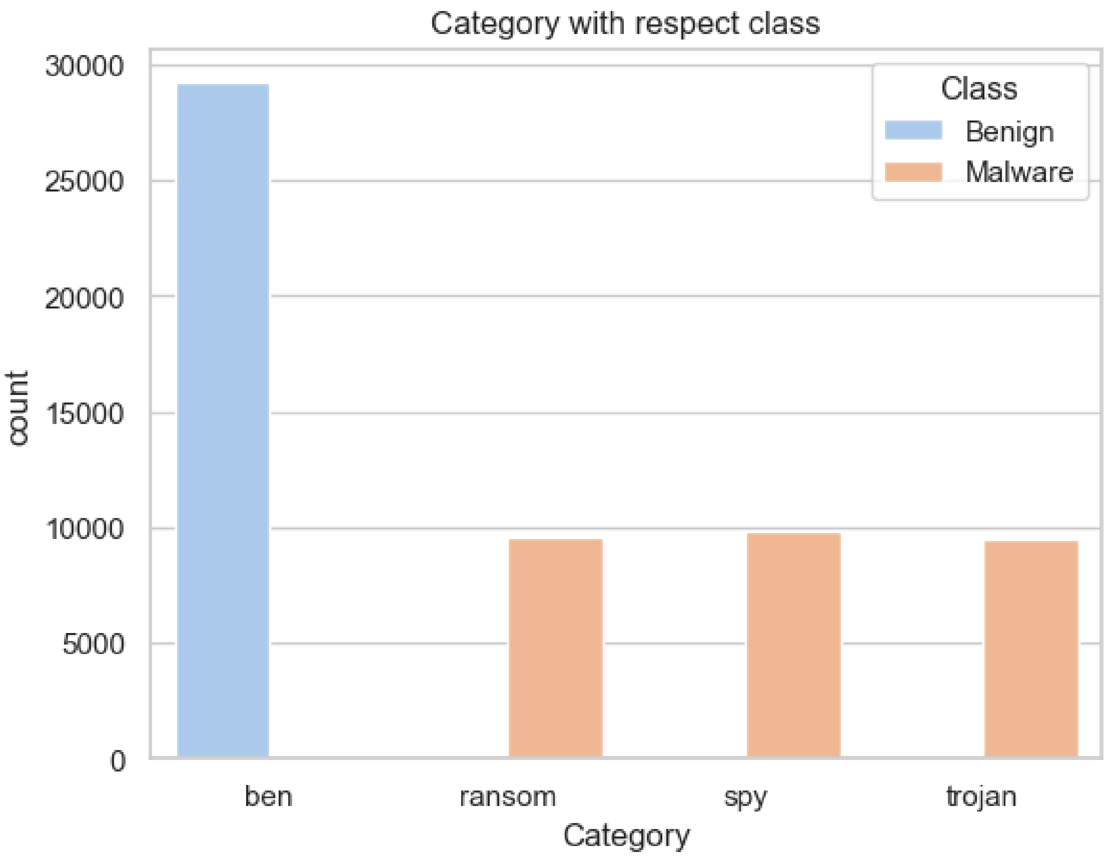

Figure 4: Comparison of Category with respect to Class

In this graph, according to the class benign and malware. Benign is higher than malwares such as ransom, spy and trojan. This can create slight imbalance in dataset as benign values are higher than that of other malwares.

5 Implementation

In the implementation phase, many machine learning, deep learning and ensemble algorithms were employed to detect memory malware. The primary models used were the MLP (Multi-Layer Perceptron) Classifier, Adaboost Classifier, Gaussian Naive Bayes Classifier, Bagging Classifier, and SGD (Stochastic Gradient Descent) Classifier. Using Jupyter Notebook 7, we train the machine learning models.

5.1 Dataset

The CIC-MalMem-2022 dataset was used in this research. This dataset was publicly available by Canadian Institute for Cybersecurity in 2022. This dataset is used for detecting obfuscated malware using memory dump. This is balanced dataset which consists of 58,596 records. Out of which 29,298 are benign and 29,298 are malicious. This dataset comprises of malwares such as Spyware, Ransomware and Trojan Horse malware. This dataset contains 57 attributes that has traces of different malwares in its memory.

Total Records	59596
Benign	29231
Ransomware	9523
Spyware	9803
Trojan	9480

Table 2: Overview of the Dataset

The dataset was preprocessed which includes handling missing values, encoding categorical variables and scaling numerical features. Then the dataset was saved in suitable format. Trained on multiple machine learning models MLP Classifier, Adaboost Classifier, Gaussian Naive Bayes Classifier, Bagging Classifier, and SGD Classifier.

5.2 Comparison of Models

MLP Classifier: The model was trained on the preprocessed dataset using the configured hyperparameters. It involved optimizing the weights and biases in the neural network through backpropagation. During training, the model learned to capture patterns and relationships within the data.

Alpha	Batch size	Solver
0.1	50	Adam

Table 3: Parameter of MLP Classifier

Adaboost Classifier: The AdaBoost model was trained on the preprocessed dataset, focusing on iteratively improving its performance by adjusting the weights of misclassified instances. Each weak learner (decision stump) was sequentially added to form a strong classifier.

Algorithm	Learning rate	Estimators
SAMME.R	0.1	50

Table 4 Parameters of AdaBoost Classifier

Gaussian Naive Bayes : The GNB model was trained on the preprocessed dataset using the Gaussian probability density function. This probabilistic model allowed GNB to make predictions based on the likelihood that a particular instance belongs to a certain class. GridSearch Approach was employed to explore values of var_smoothing during the hyperparameter tuning phase.

Var_smoothing
1e.09

Table 5: Parameters of Gaussian Naïve Bayes

Bagging Classifier: The Bagging Classifier, an ensemble learning method, was employed as a key component in the memory malware detection pipeline. Bagging, short for Bootstrap Aggregating, involves training multiple instances of a base model on different subsets of the training data to improve overall performance and reduce overfitting. The primary hyperparameters considered for the Bagging Classifier were n_estimators and max_features. n_estimators determine the number of base estimators, and max_features controls the maximum number of features considered for individual base models. A grid search was performed with values [50, 20, 10] for n_estimators and [30, 40, 50] for max_features to find the optimal combination. During the training phase, multiple base models were trained on bootstrapped subsets of the training data. The ensemble then aggregated their predictions to make the final classification.

Max_features	N_estimators
50	50

Table 6: Parameters of Bagging Classifier

Stochastic Gradient Boosting (SGB): The Stochastic Gradient Boosting (SGB) Classifier, an ensemble learning method, was employed as a powerful algorithm in the memory malware detection pipeline. SGB builds a sequence of weak learners, typically decision trees, with each learner compensating for the weaknesses of the previous one. The optimal hyperparameters determined through the grid search were loss = 'deviance', learning rate = 0.1, n_estimators = 100, and max_depth = 3. These values were chosen based on their ability to strike a balance between model complexity and generalization.

Alpha	Penalty
0.1	None

Table 7: Parameters of Stochastic Gradient Boosting

5.3 Comparison of Models

MLP and AdaBoost:
In this section, an experiment of hybrid model of MLP and Adaboost is presented. Both are algorithms are trained separately and then combined. This model was created using by combining MLP and AdaBoost model using voting mechanism based on hard or soft with cross validation as 2.

AdaBoost and Gaussian Naïve Bayes:
In this section, an experiment of hybrid model of AdaBoost and Gaussian Naive Bayes for Memory Malware detection. Hyperparameters tuning is based on GridSearchCV with parameters same as what was trained during individual algorithms, cross validation as 2 is same for all models that is trained.

Bagging and Gaussian Naive Bayes:
In this section, an experiment of hybrid model of Bagging and Gaussian Naive Bayes for Memory Malware detection. Voting classifier is based on soft or hard and hyperparameters for Bagging was set max_features= 40, n_estimators= 50 and for Gaussian Naïve Bayes var_smoothing= 1e-09 is set.

Bagging and Stochastic Gradient Descent: In this section, an experiment of combination of Bagging and Stochastic Gradient Descent model for implemented for memory malware detection. Both the algorithms were trained separately and then combined using voting scheme.

SGD and MLP: In this experiment, Stochastic Gradient Descent (SGD) and Multi-layer Perceptron (MLP) to create a hybrid model for classification. The parameter for SGD is set to alpha= 0.1, penalty= 'elasticnet' and MLP alpha= 0.001, batch_size= 50, solver= 'adam' are set.

MLP, Adaboost & Gaussian NB: In this experiment, hybrid model of MLP, Adaboost & Gaussian NB is implemented. MLP, AdaBoost and Gaussian NB are trained separately and combined with same parameters as before.

Gaussian NB, Bagging & SGD: In this experiment, hybrid model of Gaussian NB, Bagging & SGD is implemented. The training of this model is like other hybrid models.

6 Evaluation

6.1 Accuracy

The overall performance of the proposed model evaluated using accuracy. It is one of the evaluation metrics for the classification model.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

6.1.1 Accuracy for Indivdual Models

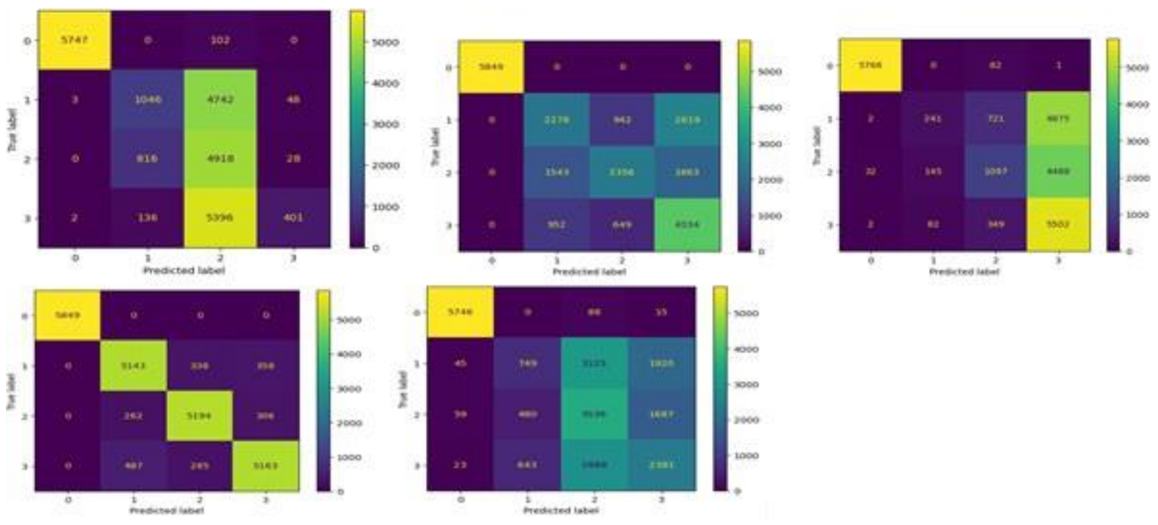

Figure 5: Visualisation for MLP, AdaBoost, Gaussian Naïve Bayes Bagging and SGB

Models Used	Accuracy	Testing time
MLP Classifier	52	0.257
AdaBoost Classifier	63	0.345
Gaussian Naïve Bayes	54	0.126
Bagging Classifier	92	0.573
SGB Classifier	53	0.078

Table 8: Accuracy of Models

These are the accuracy of five models MLP, Adaboost, Gaussian Naive Bayes [19], Bagging and SGB. By comparing all these models, Bagging proved to be best fit compared to other models. It showcased an accuracy of 92%. This model is not overfitting the training data. Both training and testing sets accuracy values are similar.

6.1.2 Accuracy for Hybrid Models

MLP and AdaBoost: The experimental results show that MLP + Adaboost to give 64% accuracy in detecting malwares. This is an average model. Grid search was employed to find the optimal hyperparameters for the ensemble model. The hyperparameters considered were related to the voting strategy and individual model parameters.

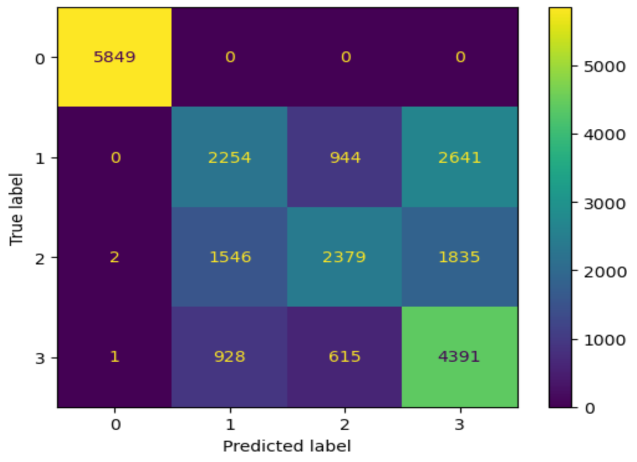

Figure 6: Confusion Matrix for MLP and Adaboost

AdaBoost and Gaussian Naive Bayes: The model accuracy is 64% which is similar to AdaBoost and MLP. This combination showcases a promising balance between accuracy and efficiency in identifying memory malware.

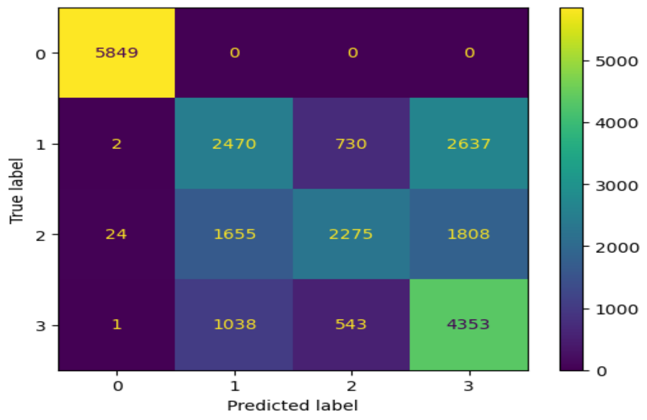

Figure 7: Confusion Matrix AdaBoost and Gaussian Naive Bayes

Bagging and Gaussian Naive Bayes: Using default hyperparameters tuning, model accuracy is 89% which is best performing hybrid model when compared to others models. This combination showcases a promising balance between accuracy and efficiency in identifying memory malware.

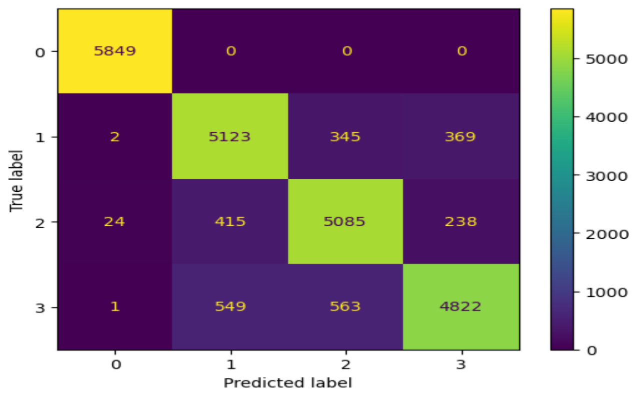

Figure 8: Confusion Bagging and Gaussian Naive Bayes for Testing

Bagging and Stochastic Gradient Descent: After tuning the hyperparameters, model accuracy showed an accuracy of 83%, precision of 85% which tells the model is performing great.

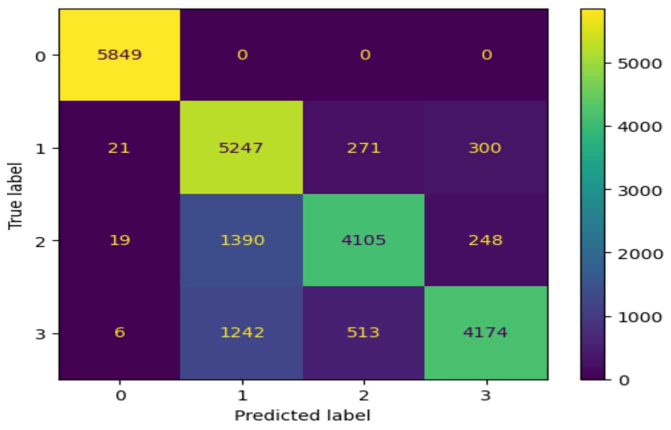

Figure 9:Visualization of Bagging and Stochastic Gradient Boosting

SGD and MLP: In this experiment, Stochastic Gradient Descent (SGD) and Multi layer Perceptron (MLP) to create a hybrid model for classification. The model achieves an accuracy of 50% which shows that performance of the model is poor compared to other models.

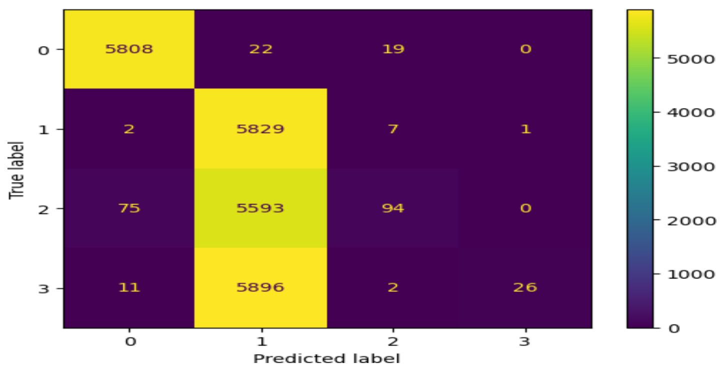

Figure 10: Visualization of SGB and MLP

MLP, Adaboost & Gaussian NB: After tuning the hyperparameters, validating and testing model it shows a accuracy of 59% with testing time of 0.55s, precision of 0.59, and F1 score of 0.56 which shows that even this model is also performing poor**.**

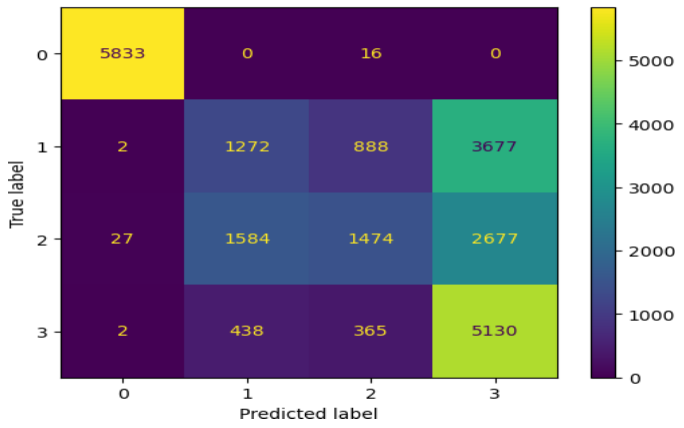

Figure 11: Visualization of MLP, AdaBoost and GNB

Gaussian NB, Bagging & SGD: After tuning the hyperparameters, validating and testing model it shows an accuracy of 59% with testing time of 0.49s, precision of 0.71, recall of 0.59 and F1 score of 0.55 which shows that even this model is also performing poor.

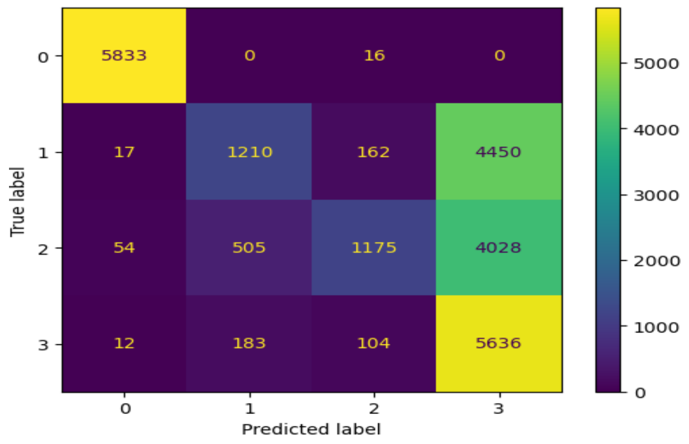

Figure 12: Visualization of Gaussian NB, Bagging & SGD

Predicted outcomes for the classification are summarised in the confusion matrix.

Hybrid Models	Accuracy
MLP and Adaboost	64
AdaBoost + Gaussian Naïve Bayes	64
Bagging and Gaussian Naive Bayes	89
Bagging and Stochastic Gradient Descent	83
SGD and MLP	50
MLP, AdaBoost & Gaussian	59

NB	
Gaussian NB, Bagging &SGD	55

Table 9: Accuracy of Hybrid Model.

6.5 Discussion

The experiments conducted in for individual classification algorithm Bagging proved to be efficient with 0.57 testing time in detecting malware. Although SGB classifier executed in 0.07s but accuracy is 53%. By prioritising higher accuracy and bit more testing time Bagging model is considered. The experiments showed that the hybrid model of Bagging and Gaussian Naive Bayes is the best performing model for memory malware detection. This model achieved an accuracy of 89%, which is significantly higher than the accuracy of any of the individual models. The hybrid model is also the most efficient, with a testing time of 1.27 seconds. The experiment was well-designed, and the results were statistically significant. However, the experiment could be improved by using a larger dataset and by using more rigorous hyperparameter tuning. The results are consistent with previous research on hybrid models for malware detection. But when comparing to individual algorithm Bagging is good model compared to other models.

7 Conclusion and Future Work

The experiments conducted to detect memory malware yielded insightful findings that underscore the strengths and considerations in employing machine learning models and ensembles for cybersecurity. Bagging Classifier showed an accuracy of 91% with time gap for testing 0.5736s. Notably, the ensemble of Bagging and Gaussian Naive Bayes exhibited commendable accuracy of 89%, emphasizing the synergy achieved through combining boosting and probabilistic modelling. According to the research conducted ensemble methods proved to be efficient and then other individual classification models and even in combination of models. The experiments also shed light on the crucial factor of time efficiency, with the ensemble demonstrating competitive accuracy while requiring less training time compared to individual models. However, limitations, such as fixed dataset size, call for future investigations with more extensive and diverse dataset. Further studies can also include in detecting subcategories of ransomware, trojan horse and spyware.

References

1. Aisha Ali-Gombe , Sneha Sudhakaran , Ramyapandian Vijayakanthan, Golden G. Richard III, cRGB_Mem: At the intersection of memory forensics and machine learning, https://doi.org/10.1016/j.fsidi.2023.301564, 7 July 2023

2. Dener, M., Ok, G., & Orman, A. (2022). Malware detection using memory analysis data in big data environment. *Applied Sciences*, *12*(17), 8604.

3. Shah, S. S. H., Ahmad, A. R., Jamil, N., & Khan, A. U. R. (2022). Memory forensics-based malware detection using computer vision and machine learning. Electronics, 11(16), 2579.

4. Alkahtani, H., & Aldhyani, T. H. (2022). Artificial intelligence algorithms for malware detection in android-operated mobile devices. Sensors, 22(6), 2268.

5. Bozkir, A. S., Tahillioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. Computers & Security, 103, 102166.

6. Noor, B., & Qadir, S. (2023). Machine Learning and Deep Learning Based Model for the Detection of Rootkits Using Memory Analysis. Applied Sciences, 13(19), 10730.

7. Xu, Z., Ray, S., Subramanyan, P., & Malik, S. (2017, March). Malware detection using machine learning based analysis of virtual memory access patterns. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017* (pp. 169-174). IEEE.

8. Klaib, A. K., Al-Nabhan, M., & Abu Al-Haija, Q. (2023, February). Identifying Memory Dump Malware Using Supervised Learning. In *Proceedings of Third International Conference on Sustainable Expert Systems: ICSES 2022* (pp. 1009-1020). Singapore: Springer Nature Singapore.

9. Bruna Moralejo, L. (2023). Machine Learning for malware detection and classification (Master's thesis, Universitat Politècnica de Catalunya).

10. Roy, K. S., Ahmed, T., Udas, P. B., Karim, M. E., & Majumdar, S. (2023). MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intelligent Systems with Applications, 20, 200283.

11. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," in IEEE Access, vol. 7, pp. 46717-46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

12. Tian, D., Zhao, R., Ma, R., Jia, X., Shen, Q., Hu, C., & Liu, W. (2022). MDCD: A malware detection approach in cloud using deep learning. Transactions on Emerging Telecommunications Technologies, 33(11), e4584.

13. Vashishtha, L. K., Chatterjee, K., & Rout, S. S. (2023). An Ensemble approach for advance malware memory analysis using Image classification techniques. Journal of Information Security and Applications, 77, 103561.

14. Gupta, D., & Rani, R. (2020). Improving malware detection using big data and ensemble learning. Computers & Electrical Engineering, 86, 106729.

15. Khalid, O., Ullah, S., Ahmad, T., Saeed, S., Alabbad, D.A., Aslam, M., Buriro, A. and Ahmad, R., 2023. An insight into the machine-learning-based fileless malware detection. Sensors, 23(2), p.612.

16. Li, H., Zhan, D., Liu, T. and Ye, L., 2019, November. Using deep-learning-based memory analysis for malware detection in cloud. In 2019 IEEE 16th international conference on mobile ad hoc and sensor systems workshops (MASSW) (pp. 1-6). IEEE

17. Naeem, H., Dong, S., Falana, O.J. and Ullah, F., 2023. Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification. Expert Systems with Applications, 223, p.119952.

18. Zelinka, I. and Amer, E., 2019, December. An ensemble-based malware detection model using minimum feature set. In Mendel (Vol. 25, No. 2, pp. 1-10).

19. Sihwail, R., Omar, K., Zainol Ariffin, K.A. and Al Afghani, S., 2019. Malware detection approach based on artifacts in memory image and dynamic analysis. Applied Sciences, 9(18),

20. Saad, S., Briguglio, W. and Elmiligi, H., 2019. The curious case of machine learning in malware detection. arXiv preprint arXiv:1905.07573.