DOI 10.15622/ia.24.1.5

Y. IMAMVERDIYEV, E. BAGHIROV, I.J. CHUKWU

DETECTING OBFUSCATED MALWARE INFECTIONS ON WINDOWS USING ENSEMBLE LEARNING TECHNIQUES

Imamverdiyev Y., Baghirov E., Chukwu I.J. **Detecting Obfuscated Malware Infections on Windows Using Ensemble Learning Techniques.**

Abstract. In the internet and smart devices era, malware detection has become crucial for system security. Obfuscated malware poses significant risks to various platforms, including computers, mobile devices, and IoT devices, by evading advanced security solutions. Traditional heuristic-based and signature-based methods often fail against these threats. Therefore, a cost-effective detection system was proposed using memory dump analysis and ensemble learning techniques. Utilizing the CIC-MalMem-2022 dataset, the effectiveness of decision trees, gradient-boosted trees, logistic Regression, random forest, and LightGBM in identifying obfuscated malware was evaluated. The study demonstrated the superiority of ensemble learning techniques in enhancing detection accuracy and robustness. Additionally, SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) were employed to elucidate model predictions, improving transparency and trustworthiness. The analysis revealed vital features significantly impacting malware detection, such as process services, active services, file handles, registry keys, and callback functions. These insights are crucial for refining detection strategies and enhancing model performance. The findings contribute to cybersecurity efforts by comprehensively assessing machine learning algorithms for obfuscated malware detection through memory analysis. This paper offers valuable insights for future research and advancements in malware detection, paving the way for more robust and effective cybersecurity solutions in the face of evolving and sophisticated malware threats.

Keywords: malware detection, machine learning, malware analysis, cybersecurity.

1. Introduction. In the rapidly evolving digital era, malware continues to be a severe and prevalent threat to computer systems worldwide. Windows operating systems, in particular, are frequent targets due to their extensive user base and the variety of vulnerabilities that malicious actors can exploit. Malware can lead to severe consequences, including data theft, system damage, and financial loss, necessitating robust detection mechanisms to safeguard systems and users. The ongoing battle between malware distributors and the extensive efforts mobilized for malware detection persists, driven by the destructive potential of malware. This includes significant financial losses, disruption of critical services, and even human casualties in critical infrastructures such as SCADA. Cybercriminals leverage malware as a weapon due to its capacity to inflict widespread harm and chaos [1].

Traditional signature-based methods, which rely on identifying known malware signatures, have proven insufficient in the face of new and sophisticated malware variants. These methods often fail to detect novel threats and zero-day attacks that lack predefined signatures. As a result, the focus has shifted

towards machine learning and behavioral-based detection approaches, which offer the potential to identify previously unseen malware by analyzing patterns and behaviors indicative of malicious intent [2]. This shift is driven by the increasing complexity and obfuscation techniques employed by malware authors, making static analysis methods less effective [3].

On the other hand, dynamic analysis involves executing the software in a controlled environment, such as a sandbox, to observe its behavior and interactions with the system. This method can detect malware that evades static analysis by monitoring runtime behavior, including network activity, file modifications, and registry changes. The strength of dynamic analysis lies in its ability to identify zero-day threats and polymorphic malware. However, it is resource-intensive and time-consuming, requiring a secure environment to execute potentially harmful software. Additionally, sophisticated malware can detect when it is running in a sandbox and alter its behavior to avoid detection, reducing the effectiveness of dynamic analysis [4].

Machine learning-based malware detection leverages the power of statistical analysis and pattern recognition to detect malicious activities in real time. Techniques such as ensemble methods, which combine multiple machine learning models, have shown significant promise in enhancing detection accuracy and robustness. By integrating the strengths of various base classifiers, ensemble techniques can improve detection performance and reduce false positives, making them a valuable tool in the fight against malware [5]. Recent studies have demonstrated the effectiveness of various machine learning and deep learning approaches in malware detection, highlighting the need for continual advancement in this field [6–8].

Moreover, profound and self-supervised learning advancements have opened new avenues for malware detection. Techniques such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown potential in identifying complex patterns within large datasets, improving detection rates for previously unseen malware [4]. Self-supervised learning approaches, which do not require large labeled datasets, offer promising solutions for developing efficient and scalable malware detection systems [9].

Despite these advancements, challenges remain in deploying machine learning-based malware detection systems. Issues such as model interpretability, adversarial attacks, and the need for large labeled datasets must be addressed to ensure the effectiveness and reliability of these systems in real-world scenarios. Research efforts continue to explore innovative solutions to these challenges, aiming to develop more robust and adaptable malware detection frameworks [10, 11].

The primary objectives and contributions of this paper can be summarized as follows:

Interpretability of Model Predictions: To enhance the interpretability of our model predictions, we employ SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). These techniques elucidate the contribution of various system and process-related features to the model predictions, improving transparency and trustworthiness.

Identification of Key Features: Our study identifies vital features that significantly impact malware detection, such as the number of process services, active services, file handles, registry keys, and callback functions. Understanding these features helps refine detection strategies and improve model accuracy.

Future Research Directions: The paper outlines future research directions, including exploring deep learning models, real-time detection systems, dataset expansion, and the integration of behavioral analysis. These directions aim to advance the field of malware detection further and enhance the robustness of detection systems.

The remainder of this paper is structured as follows: Section 3 reviews related works in the field of malware detection, discussing previous studies and their methodologies. Section 4 details the dataset and the machine learning models used in the analysis. In contrast, section 5 presents the results, including the performance metrics of different models and the explainability results using SHAP and LIME to understand the model predictions. Finally, section 6 describes the conclusion and outlines future research directions, summarizing the essential findings and suggesting areas for further investigation.

2. Problem statement. Malware continues to pose a significant threat to Windows operating systems, exploiting the extensive user base and diverse vulnerabilities. Traditional heuristic and signature-based detection methods are increasingly inadequate against sophisticated threats, particularly obfuscated malware designed to evade detection by altering its appearance and behavior. This study aims to develop a robust, cost-effective system for detecting obfuscated malware using memory dump analysis and ensemble learning techniques. By evaluating machine learning algorithms on the CIC-MalMem-2022 dataset and employing SHAP and LIME for model interpretability, this research seeks to enhance detection accuracy, robustness, and transparency in malware detection.

The study utilizes ensemble learning techniques, including methods such as Random Forest, Gradient Boosting, and LightGBM, known for their robustness in handling complex datasets and their ability to generalize well across various conditions. Robustness is ensured through the use of ensemble

methods, which reduce the likelihood of overfitting by aggregating predictions from multiple models, thus enhancing the stability and reliability of the system. Cost-effectiveness is a key consideration in the selection of models and the design of the system. By leveraging memory dump analysis – a method that focuses on analyzing snapshots of system memory – we reduce the computational overhead associated with real-time monitoring and analysis. Additionally, the choice of LightGBM, known for its efficiency in both training time and memory usage, further contributes to the cost-effectiveness of the system. This approach allows for the deployment of the detection system in environments with limited computational resources, making it practical for widespread use.

3. Related work. Several studies, drawing from IEEE Xplore, Web of Science, and Scopus databases,, have focused on applying machine learning and deep learning techniques in the field of malware detection and adversarial attack mitigation [12–32]. In particular, some studies focus on leveraging machine learning models to enhance malware detection capabilities while addressing the vulnerabilities posed by adversarial attacks [13, 15, 17, 18, 20, 22, 23, 26, 27, 29–31]. These efforts include the development of frameworks like EvadeDroid [12], which applies a practical evasion attack on Android malware detectors, and MEME [13], a model-based reinforcement learning algorithm designed to create adversarial malware capable of bypassing detection systems. Additionally, other works have explored the use of deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to classify malware based on various features, including API call sequences and syscall subsequences, as seen in studies focusing on these methods [18–20, 22, 31]. These approaches not only demonstrate the effectiveness of machine learning in identifying malware but also highlight the challenges posed by adversarial examples, necessitating the development of robust defense mechanisms, such as adversarial training and randomized smoothing techniques, to safeguard these systems.

A novel opcode-based methodology that leverages multiple behavioral target variables to enhance static malware classification was proposed by the authors in [6]. Their methodology's robustness against random opcode injection attacks was validated on the AMDArgus and MOTIF datasets, achieving superior mean classification accuracy and F1 scores compared to other convolution-based architectures. While the proposed opcode-based malware classification approach shows promise, it also has some limitations. One significant drawback is the assumption that all weak target variables are independent, neglecting the potentially complex relationships between them.

This simplification might limit the model's effectiveness in capturing nuanced patterns in the data.

A novel malware detection scheme for Smart IoT environments called Mal3S, which leverages a multi-spatial pyramid pooling network, was suggested by the authors in [3].Their approach involves static analysis to extract features such as bytes, opcodes, API calls, strings, and dynamic link libraries (DLLs). These are then converted into images of different sizes for training the SPP-net model. Evaluating Mal3S on three malware datasets, they achieved an average detection accuracy of 98.02% and a classification accuracy of 98.43%, outperforming existing techniques. This method also demonstrated effective generalization capabilities across different types of malware. However, the approach's reliance on static analysis means it might struggle to detect malware that extensively obfuscates its code or dynamically modifies its behavior. Future work could explore integrating dynamic analysis techniques to address these limitations and enhance detection accuracy.

The approach for malware classification using self-supervised learning, named MalSSL, was suggested in [4], addressing the challenges of requiring large labeled datasets. MalSSL utilizes image representation, contrastive learning, and data augmentation to classify malware without needing labeled data. The model is first trained on an unlabeled Imagenette dataset as a pretext task and then retrained on an unlabeled malware dataset for downstream tasks, including malware family and benign classification. The results show an accuracy of 98.4% for the malware family classification on the Malimg dataset and 96.2% for the malware and benign classification on the Maldeb dataset, outperforming other self-supervised methods. However, the reliance on pretraining with a dataset like Imagenette might limit its adaptability to more diverse or complex malware datasets. Future work could explore enhancing the model's adaptability and testing its efficacy on varied and evolving malware datasets.

A comparative performance analysis of malware detection algorithms based on various texture features and classifiers was suggested by the authors in [10] to address challenges. Their method includes four stages: converting malware to grayscale, extracting features using segmentation-based fractal texture analysis (SFTA), Local Binary Pattern (LBP), Haralick, Gabor, and Tamura, classifying with Gaussian Discriminant Analysis (GDA), k-Nearest Neighbor (KNN), Logistic, Support Vector Machines (SVM), Random Forest (RF), and Extreme Learning Machine (Ensemble), and evaluating the results. The study used the Malimg imbalanced and MaleVis balanced datasets to assess classifier performance and feature effectiveness. Results indicated that KNN outperformed other classifiers in accuracy, error, F1, and precision, with

SVM and RF as runners-up. Gabor performed better in MaleVis, while SFTA excelled in the Malimg dataset. The SFTA-KNN and Gabor-KNN methods achieved 96.29% and 98.02% accuracy, respectively, surpassing current state-of-the-art approaches. However, the study relied on specific feature extraction methods and comparative analysis using balanced and imbalanced datasets, revealing that balanced datasets significantly improved accuracy and precision while reducing error compared to imbalanced datasets.

The application of several machine-learning algorithms to build a malware detection model for Android systems was suggested by the authors in [11]. Traditional methods of detecting malware using anti-virus software often fall short due to the rapid increase in applications and potentially embedded advertisements or unwanted software. To address this, the authors developed unweighted and weighted models to handle unbalanced data. Their experiments indicated that the weighted random forest model achieved the best performance with an accuracy of 98.94%. However, the study primarily focuses on static analysis and may not account for dynamically changing malware behaviors. Future research could explore incorporating dynamic analysis techniques to enhance detection capabilities further.

A cost-effective obfuscated malware detection system, utilizing diverse machine-learning algorithms through memory dump analysis, was proposed by the authors in [33]. The research focused on the CIC-MaIMem-2022 dataset, simulating real-world scenarios to evaluate the effectiveness of decision trees, ensemble methods, and neural networks in detecting obfuscated malware. Despite the balanced nature of the dataset, with equal malware and benign samples (50%), the authors highlight the application of undersampling and oversampling methods to address potential imbalances within specific malware categories. However, in real-world scenarios, these methods often do not have a positive impact, as they can lead to overfitting or underfitting, reducing the model's generalizability.

Common problems in malware detection research include the over-reliance on static analysis, which struggles against malware with dynamic behavior or advanced obfuscation techniques, and the frequent issue of imbalanced datasets that lead to overfitting or underfitting in models. Simplifying complex relationships between features by assuming their independence can result in models that miss intricate patterns, reducing classification accuracy. Additionally, the use of specific feature extraction methods may limit the generalizability of models to different types of malware. Pretraining with non-malware-specific datasets further hampers adaptability, underscoring the need for integrating dynamic analysis techniques, enhancing

dataset diversity, and capturing interdependencies more effectively in future research.

4. Methodology. Dataset description. The CIC-MalMem-2022 dataset [34], used for this study, comprises memory dumps categorized into four classes: benign, spyware, ransomware, and trojan. Detailed memory features such as process counts, threads, handles, and DLLs are extracted, which help identify malicious patterns. The dataset is balanced, with a total of 58,596 records. Specifically, it includes 29,298 benign records (50%), 10,020 spyware records (17.1%), 9,791 ransomware records (16.7%), and 9,487 trojan records (16.2%).

It is important to note that this is a multiclass classification problem rather than a binary classification task. The distribution of classes reflects a realistic scenario where benign processes are more common, while the various types of malware are less prevalent. This class imbalance (50% benign and the remaining 50% distributed among the three types of malware) adds complexity to the classification task and aligns with real-world situations where malware constitutes a smaller, yet significant, portion of system processes.

Enviromental setup. Our analysis and modeling experiments were conducted using the robust Dataiku platform with advanced data analytics and machine learning capabilities. Dataiku offers a comprehensive suite of tools for data preparation, feature engineering, model development, and evaluation, making it an ideal environment for our research. For this study, we utilized Dataiku version 10.0.5 (licensed), with the notebook server running version 5.4.0-dku10.0-0 and Python 3.6.8.

Model description. In this section, we delve into the machine learning models employed in this study, examining their fundamental principles, loss functions, activation functions, and mathematical formulations. We also discuss each model's strengths, weaknesses, and limitations, providing a comprehensive understanding of their capabilities in the context of obfuscated malware detection.

Decision Tree. Decision Trees are used for classification tasks by recursively splitting the data into subsets based on input feature values. The split criterion, typically Gini impurity or entropy, evaluates the quality of splits. Gini impurity is shown in Equation 1.

$$Gini = \sum_{i=1}^{n} p_i(1 - p_i). \tag{1}$$

In this formula, n represents the total number of classes, and p_i is the probability of an element being classified to class i.

Entropy is shown in Equation 2:

$$Entropy = -\sum_{i=1}^{n} p_i \log(p_i). \tag{2}$$

Strengths include interpretability and ease of implementation, while weaknesses involve susceptibility to overfitting and poor performance on complex datasets [35].

Gradient Boosted Trees. Gradient Boosting builds models sequentially, correcting the errors of previous models. It minimizes a specified loss function using gradient descent. The loss function is shown in Equation 3:

$$L_m(y, F(x)) = \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + v \cdot h_m(x_i)). \tag{3}$$

This formula represents the loss function used in gradient boosting, where $L_m(y, F(x))$ is the loss for each instance i, y_i is the actual value, $F_{m-1}(x_i)$ is the prediction from the previous iteration, v is the learning rate, and $h_m(x_i)$ is the new model to be added.

Strengths include high accuracy and robustness against overfitting, while weaknesses include longer training times and complexity in tuning hyperparameters [36].

LightGBM. LightGBM (Light Gradient Boosting Machine) is designed for speed and performance, using a histogram-based approach to find the best-split points, reducing memory usage and increasing training speed. Histogram-based decision tree learning is shown in Equation 4:

$$G = \sum_{i=1}^{n} \frac{g_i^2}{h_i}, \tag{4}$$

where n is the number of instances, g_i is the gradient of the loss function concerning the prediction, for example, i, and h_i is the Hessian (second derivative) of the loss function concerning the prediction for instance i.

Strengths include high accuracy, scalability, and efficiency, while weaknesses might involve sensitivity to hyperparameter settings [37].

Logistic Regression. Logistic Regression models the probability of a binary classification by applying the logistic function to a linear combination

of input features. The logistic function has the following form:

$$P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_n x_n)}}, \tag{5}$$

where $P(y=1|x)$ is the probability of the binary outcome y being one given the input features x. The expression $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ is a linear combination of the input features $x_1, x_2, \ldots, x_n$ with their respective coefficients $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$. The logistic function (sigmoid function) transforms this linear combination into a probability value between 0 and 1.

Strengths include simplicity and interpretability, while weaknesses involve limitations in handling non-linear relationships [38].

Random Forest. Random Forest constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (Regression) of the individual trees. The Random Forest algorithm is shown in Equation 6:

$$\dot{f}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x). \tag{6}$$

Here, $\dot{f}(x)$ is the final prediction, B is the number of individual models in the ensemble, and $f_b(x)$ is the prediction of the b-th individual model. The ensemble prediction is obtained by averaging the predictions of all individual models.

Strengths include robustness and reduced overfitting, while weaknesses involve complexity and longer training times for large datasets [35].

XGBoost. XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting library for high efficiency, flexibility, and portability. It uses a more regularized model formalization to control overfitting. The regularized objective is shown in Equation 7:

$$L(\theta) = \sum_{i=1}^{n} l(y_i, \dot{y_i}) + \sum_{k=1}^{K} \Omega(f_k). \tag{7}$$

In this formula, $L(\theta)$ represents the total loss, where the first term $\sum_{i=1}^{n} l(y_i, \dot{y_i})$ accounts for the loss for each instance i, with y_i being the true value and $\dot{y_i}$ being the predicted value. The second term $\sum_{k=1}^{K} \Omega(f_k)$ is the regularization term that penalizes the complexity of the model, where $\Omega(f_k)$ applies to each feature k to prevent overfitting.

Strengths include high performance and efficiency, while weaknesses involve complexity in implementation and tuning [39].

Evaluation metrics. To comprehensively understand our models' performance and robustness, we evaluated using these metrics:

Accuracy. Accuracy measures the proportion of correctly predicted instances out of the total cases. While accuracy provides a general performance measure, it may not be suitable for imbalanced datasets.

Precision. Precision, also known as the positive predictive value, indicates the proportion of accurate positive predictions out of all positive predictions. Precision is crucial when the cost of false positives is high.

Recall. Recall, also known as sensitivity or actual positive rate, measures the proportion of accurate positive predictions out of all actual positive instances. Recall is necessary when the cost of false negatives is high.

F1-Score. The F1-score is the harmonic mean of precision and recall, providing a balanced measure of both metrics. It is beneficial when dealing with imbalanced datasets.

ROC-AUC. The ROC-AUC score evaluates the model's ability to discriminate between positive and negative classes. The ROC curve plots the actual positive rate (recall) against the false positive rate. The AUC represents the area under this curve, with a value closer to 1 indicating better model performance.

The evaluation metrics have been shown in Table 1, where the following formulas are used.

Table 1. Metrics used

Metric	**Formula**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN} \times 100$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-Score	$2 \times \frac{Precision \times Recall}{Precision+Recall}$

Explainability. By incorporating explainability into our methodology, we ensured that our machine-learning models for detecting obfuscated malware are transparent and interpretable. While decision trees, regression models, and ensemble methods are generally considered interpretable, the level of interpretability can vary significantly depending on the specific model and its complexity. Simple models like linear regression or single decision trees offer straightforward interpretations; their predictions can be easily understood

by examining coefficients or the structure of the tree, respectively. However, as models become more complex – particularly with the use of ensemble techniques such as boosting and bagging – their interpretability diminishes. Ensemble methods, by their nature, involve the aggregation of predictions from multiple models, often hundreds or thousands of decision trees, each contributing to the final output.

In these complex scenarios, the simple interpretability associated with individual models becomes obscured. It is no longer practical to visualize or directly understand the contribution of each feature across all the constituent models within an ensemble. The final prediction emerges from the collective behavior of many models, making it difficult to deconstruct the prediction into understandable parts.

We used SHapley Additive exPlanations (SHAP) [40] and Local Interpretable Model-agnostic Explanations (LIME) [41] to enhance the explainability of our machine-learning models for detecting obfuscated malware. These techniques assisted us in understanding and interpreting the decisions made by complex models, ensuring transparency and trust.

SHAP (SHapley Additive exPlanations). SHAP values are based on cooperative game theory, providing a unified feature importance measure. They explain how each feature contributes to the prediction by averaging over all possible orderings of features. SHAP ensures three properties: local accuracy, missingness, and consistency. Local accuracy ensures that the sum of feature attributions matches the model output for each instance. Missingness guarantees that features not present in the model have no impact. Consistency ensures that if a model changes such that a feature's contribution increases or stays the same, the attribution should not decrease. SHAP values can be computed using various methods such as Kernel SHAP, which approximates the values for any model type [40].

The formula for the SHAP values is given in Equation 8:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{i\}) - f(S) \right], \tag{8}$$

where ϕ_i is the SHAP value for feature i, S is a subset of all features N excluding i, $f(S \cup \{i\})$ represents the prediction for the model including feature i in the subset S, and $f(S)$ represents the prediction excluding feature i. The term $\frac{|S|!(|N|-|S|-1)!}{|N|!}$ is a weighting factor based on the size of the subset.

LIME (Local Interpretable Model-agnostic Explanations). LIME explains the predictions of any classifier by approximating it locally with an

interpretable model. It perturbs the data around the instance to be presented and trains a simple, interpretable model (like linear Regression) on these perturbed samples. This local model can provide insights into how each feature influences the prediction in that particular vicinity of the instance. LIME's essence is to balance interpretability and fidelity to the original model [41].

The formula for the LIME explanation model is given in Equation 9:

$$\xi(x) = \arg\min_{g \in G} \sum_{z \in Z} \pi_x(z) \left(f(z) - g(z)\right)^2 + \Omega(g). \tag{9}$$

In this formula, $\xi(x)$ is the explanation model for the instance x, $g \in G$ represents a family of interpretable models, $\pi_x(z)$ is a proximity measure between z and x, $f(z)$ is the prediction of the complex model, and $g(z)$ is the prediction of the interpretable model. The term $\Omega(g)$ is a regularization term to ensure simplicity in the explanation model.

5. Results of the experiments. Our study applied multiple machine-learning models to the CIC-MalMem-2022 dataset to evaluate their performance in detecting obfuscated malware. The models were evaluated based on accuracy, precision, recall, F1-score, and ROC AUC. The results are summarized in Table 2.

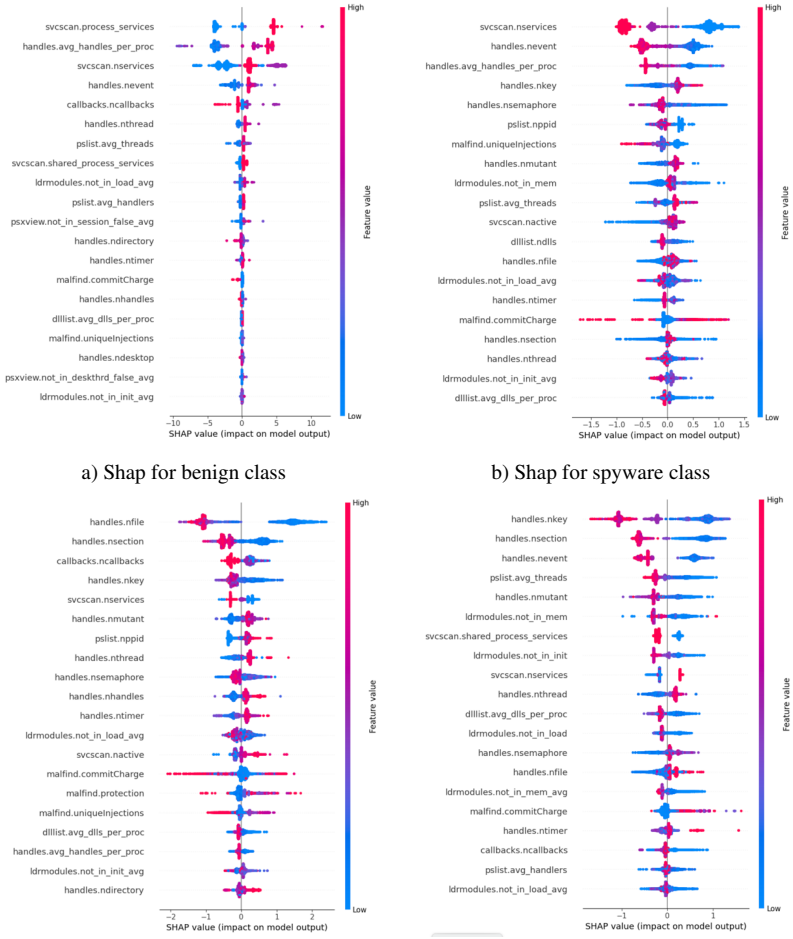
Table 2. Performance comparison of machine learning models for malware detection

Model	Train Time	Accuracy	Precision	Recall	F1-score	ROC AUC
Decision Tree	6s	0.76	0.67	0.64	0.64	0.84
Gradient Boosted Trees	23s	0.81	0.72	0.72	0.72	0.91
LightGBM	28s	0.87	0.81	0.81	0.81	0.95
Logistic Regression	1m 14s	0.74	0.62	0.61	0.61	0.83
Random Forest	1m 9s	0.87	0.80	0.80	0.80	0.95
XGBoost	21s	0.82	0.73	0.73	0.73	0.92

The performance analysis of the machine-learning models on the CIC-MalMem-2022 dataset reveals several key insights. LightGBM achieved the highest F1 score at 0.81, indicating a balanced performance in terms of precision and recall, which is notably higher than other authors' results on the same dataset, with [42] reporting an F1 score of 68% and [43] achieving 70.33%. This score demonstrates that LightGBM is highly effective in identifying true positives while minimizing false positives and false negatives. Similarly, LightGBM also achieved the highest ROC AUC score of 0.95, highlighting

its excellent capability to distinguish between benign and malicious memory dumps. The high ROC AUC score suggests that LightGBM is highly effective in distinguishing between the positive and negative classes.

SHAP explanation. The SHAP (SHapley Additive exPlanations) values for each class of malware and benign processes are presented in Figures 1(a), 1(b), 1(c), and 1(d).



a) Shap for benign class

b) Shap for spyware class

c) Shap for trojan class

d) Shap for ransomware class

Fig. 1. SHAP explanations for different classes: a) benign; b) spyware; c) trojan; d) ransomware

These figures illustrate the impact of various features on the model output for each class. After analyzing the results, we can draw several significant conclusions about the behavior and characteristics of each type of malware and benign processes.

Ransomware is characterized by high values of handles.nkey indicates that these processes heavily utilize registry keys, possibly for configuration and execution. Extensive use of memory sections (handles.nsection) suggests complex memory operations. High event handle usage indicates the reliance on synchronization mechanisms (handles.nevent). Furthermore, ransomware processes tend to have a higher average number of threads (pslist.avg_threads), indicating parallel operations and multitasking. High values of mutant handles (handles.nmutant) point to advanced process control and manipulation.

Trojans exhibit a distinctive pattern where they frequently use many file handles (handles.nfile), likely for file manipulation or monitoring activities. They also make extensive use of memory sections, similar to ransomware. Higher callback counts (callbacks.ncallbacks) suggest that trojans hook into numerous system processes to maintain control and monitor activities. Their extensive interaction with the registry, utilizing numerous registry keys (handles.nkey), is notable. The presence of many running services (svcscan.nservices) indicates that trojans might rely on system services for persistence and functionality.

Spyware processes are marked by their involvement with multiple services (svcscan.nservices), possibly for data collection and transmission. High event handle usage indicates significant synchronization operations within spyware processes. High average handle usage per process (handles.avg_handles_per_proc) suggests intensive interaction with system resources. Frequently engaging with the registry (handles.nkey) and using semaphores (handles.nsemaphore) indicate multiple concurrent processes.

Benign processes, in contrast, show a pattern of routine service operations (svcscan.process_services). Higher average handle usage per process is typical in benign software, reflecting standard interactions with system resources. Everyday system events are frequent in benign processes (handles.nevent), and the presence of callbacks is typical for maintaining standard system functionality. In summary, malicious processes (ransomware, trojan, spyware) generally use handles and registry keys more, indicating manipulation and monitoring activities. Event and memory section handles are frequently used in ransomware and spyware, suggesting complex synchronization and memory usage patterns. Service and callback usage are significant across all classes, differentiating between types of malware and benign processes.

Ransomware is identified by high thread and mutant handle usage, trojans by extensive file and section handle usage, and spyware by intensive service and handle operations. Benign processes exhibit regular service and handle usage patterns, which are typical of standard system operations.

LIME explanation. The provided LIME (Local Interpretable Model-agnostic Explanations) explanation, shown in Figure 2, visualizes the prediction of the LightGBM model for a particular instance. This instance is classified as 'Spyware' with a probability of 0.52. The LIME plot illustrates how different features contribute to the prediction, showing their impact on the model's decision.

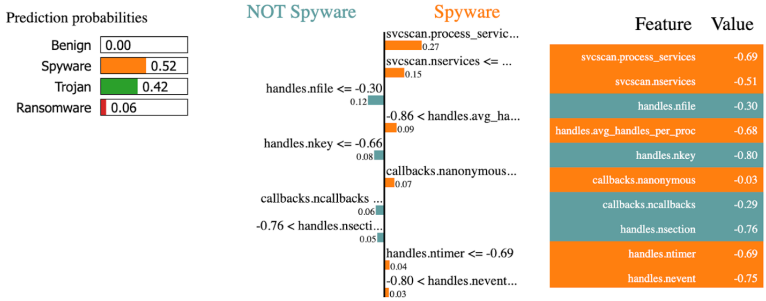


Fig. 2. LIME explanation for spyware instance

The model predicts the instance as 'Spyware' with a probability of 0.52, followed by 'Trojan' with a probability of 0.42. The probabilities for 'Benign' and 'Ransomware' are much lower, at 0.00 and 0.06, respectively. The bar graph on the right-hand side of the LIME plot lists the features and their respective values that contributed to the prediction. Features that increase the likelihood of the instance being classified as 'Spyware' are highlighted in orange, whereas features that decrease the possibility are shown in teal.

We observe that the number of process services (svcscan.process_services) had the highest positive impact on the Spyware classification, with a value of -0.69. This suggests that the number of process services is indicative of spyware activity. Similarly, the number of active services (svcscan.nservices), with a value of -0.51, also positively influenced the classification towards 'Spyware,' implying that the number of active services is a significant factor. The number of file handles (handles.nfile), with a value of -0.30, contributed positively to the classification, suggesting that spyware processes involve numerous file handles.

Further, the average number of handles per process (handles.avg_handles_per_proc), with a value of -0.68, indicates higher activity for spyware. The number of registry keys (handles.nkey), with a value of -0.80, significantly influenced the model toward predicting 'Spyware.' Additionally, anonymous callback functions (callbacks.nanonymous), with a value of -0.03, and the total number of callback functions (callbacks.ncallbacks), with a value of -0.29, also played a role in the classification. Memory-related features such as the number of memory sections (handles.nsection), with a value of -0.76, and the number of timer handles (handles.ntimer), with a value of -0.69, were also influential. The number of event handles (handles.nevent), with a value of -0.75, was another significant factor.

From the LIME explanation, it is clear that the LightGBM model relies heavily on specific system and process-related features to distinguish between different types of malware. For this particular instance, classified as 'Spyware', the number of process services, active services, file handles, and registry keys were vital indicators. In addition, the average number of handles per process, anonymous callback functions, and memory-related features were critical to the prediction.

6. Conclusion and future works. This paper demonstrates a comprehensive approach to detecting obfuscated malware through memory dump analysis using various machine-learning algorithms. Our study leveraged the CIC-MalMem-2022 dataset, which simulates real-world scenarios to evaluate the effectiveness of machine-learning models in identifying obfuscated malware. We implemented and assessed multiple algorithms, including decision trees, gradient-boosted trees, logistic regression, random forest, and LightGBM, to understand their strengths and limitations in malware detection.

The results of the study confirm that the proposed system achieves both robustness and cost-effectiveness, meeting the goals outlined at the outset. The use of ensemble learning techniques, particularly LightGBM, ensures that the system remains robust even when faced with challenging data conditions, such as obfuscated malware samples. Furthermore, the system's efficiency in terms of computational resource usage makes it cost-effective, allowing it to be deployed in environments where resources are limited. This combination of robustness and cost-effectiveness is crucial for practical applications in real-world cybersecurity scenarios, where systems must not only perform accurately but also operate efficiently.

Our findings highlight the superior performance of ensemble learning techniques, particularly LightGBM, in achieving higher detection accuracy and robustness across diverse malware types. We further enhanced the

interpretability of our models using SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME), which provided valuable insights into the contribution of various system and process-related features to the model predictions. Features such as the number of process services, active services, file handles, registry keys, and callback functions were identified as significant indicators in distinguishing between different types of malware and benign processes.

In conclusion, integrating advanced machine learning algorithms and interpretability techniques offers a promising solution to improve malware detection capabilities. This study paves the way for further research in developing robust, interpretable, and practical cybersecurity solutions to combat the ever-evolving landscape of malware threats.

Although this study provides a comprehensive approach to obfuscated malware detection using memory dump analysis and machine learning, several avenues for future research and enhancement remain. Future work could explore the application of deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which have shown promise in various complex classification tasks. Implementing real-time detection systems that can analyze memory dumps and detect malware on the fly is another crucial step. Expanding the dataset to include more diverse and recent malware samples, including those targeting different operating systems and platforms (e.g., macOS, Linux, Android, and IoT devices), would improve the generalizability of the models. Additionally, incorporating benign samples from a broader range of applications and user behaviors could further enhance the model's ability to distinguish between benign and malicious activities. Conducting a more in-depth investigation into feature engineering and selection, studying the impact of adversarial attacks, and exploring other explainable AI techniques would further improve model performance and transparency. Integrating the proposed detection system with existing security frameworks, incorporating behavioral analysis, and developing collaborative defense mechanisms where different systems share threat intelligence could enhance the overall cybersecurity landscape. Addressing regulatory and ethical considerations in the deployment of machine learning-based malware detection systems is also essential. By pursuing these future research directions, we can further advance the field of malware detection, creating more robust, efficient, and interpretable solutions to protect against the ever-evolving landscape of cyber threats.

Declaration of Generative AI and AI-assisted technologies in the writing process. While preparing this work, the authors used ChatGPT for language editing and refinement. After using this tool/service, the author

reviewed and edited the content as needed and took full responsibility for the content of the publication.

References

1. Baghirov E. Evaluating the performance of different machine learning algorithms for Android malware detection. In 2023 5th International Conference on Problems of Cybernetics and Informatics (PCI). IEEE, 2023. pp. 1–4. DOI: 10.1109/PCI60110.2023.10326006.
2. Baghirov E. Comprehensive framework for malware detection: Using ensemble methods, feature selection, and hyperparameter optimization. In 2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT). IEEE, 2023. pp. 1–5. DOI: 10.1109/AICT59525.2023.10313179.
3. Jeon J., Jeong B., Baek S., Jeong Y.-S. Static Multi Feature-Based Malware Detection Using Multi SPP-net in Smart IoT Environments. IEEE Transactions on Information Forensics and Security. 2024. vol. 19. pp. 2487–2500. DOI: 10.1109/TIFS.2024.3350379.
4. Ismail S.J.I., Hendrawan Rahardjo B., Juhana T., Musashi Y. MalSSL – Self-Supervised Learning for Accurate and Label-Efficient Malware Classification. IEEE Access. 2024. vol. 12. pp. 58823–58835. DOI: 10.1109/ACCESS.2024.3392251.
5. Baghirov E. Malware detection based on opcode frequency. Journal of Problems of Information Technology, 2023. vol. 14(1). pp. 3–7. DOI: 10.25045/jpit.v14.i1.01.
6. Egitmen A., Yavuz A.G., Yavuz S. TRConv: Multi-Platform Malware Classification via Target Regulated Convolutions. IEEE Access. 2024. vol. 12. pp. 71492–71504. DOI: 10.1109/ACCESS.2024.3401627.
7. Gungor A., Dogru I.A., Barisci N., Toklu S. Malware detection using image-based features and machine learning methods. Journal of the Faculty of Engineering and Architecture of Gazi University. 2023. vol. 38. no. 3. pp. 1781–1792. DOI: 10.17341/gazimmfd.994289.
8. Mesbah A., Baddari I., Riahla M.A. LongCGDroid: Android malware detection through longitudinal study for machine learning and deep learning. Jordanian Journal of Computers and Information Technology. 2023. vol. 9. no. 4. pp. 328–346. DOI: 10.5455/jjcit.71-1693392249.
9. Howard A., Hope B., Saltaformaggio B., Avena E., Ahmadi M., Duncan M., McCann R., Cukierski W. Microsoft Malware Prediction. Kaggle, 2018. Available at: https://kaggle.com/competitions/microsoft-malware-prediction. (accessed 26.10.2024).
10. Ahmed I.T., Hammad B.T., Jamil N.A Comparative Performance Analysis of Malware Detection Algorithms Based on Various Texture Features and Classifiers. IEEE Access. 2024. vol. 12. pp. 11500–11519. DOI: 10.1109/ACCESS.2024.3354959.
11. Xie W., Zhang X. The Application of Machine Learning in Android Malware Detection. 2024 4th International Conference on Neural Networks, Information and Communication Engineering (NNICE). 2024. pp. 1–4. DOI: 10.1109/NNICE61279.2024.10498936.
12. Bostani H.; Moonsamy V. EvadeDroid: A practical evasion attack on machine learning for black-box Android malware detection. Computers and Security. 2024. vol. 139. DOI: 10.1016/j.cose.2023.103676.
13. Rigaki M., Garcia S. The Power of MEME: Adversarial Malware Creation with Model-Based Reinforcement Learning. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2024. pp. 44–64. DOI: 10.1007/978-3-031-51482-1 _3.
14. Rudd E.M., Krisiloff D., Coull S., Olszewski D., Raff E., Holt J. Efficient Malware Analysis Using Metric Embeddings. Digital Threats: Research and Practice. 2024. vol. 5(1). pp. 1–20. DOI: 10.1145/3615669.

15. Zhan D., Zhang Y., Zhu L., Chen J., Xia S., Guo S., Pan Z. Enhancing reinforcement learning based adversarial malware generation to evade static detection. Alexandria Engineering Journal. 2024. vol. 98. pp. 32–43. DOI: 10.1016/j.aej.2024.04.024.
16. Aljabri M., Alhaidari F., Albuainain A., Alrashidi S., Alansari J., Alqahtani W., Alshaya J. Ransomware detection based on machine learning using memory features. Egyptian Informatics Journal. 2024. vol. 25. DOI: 10.1016/j.eij.2024.100445.
17. Ban Y., Kim M., Cho H. An Empirical Study on the Effectiveness of Adversarial Examples in Malware Detection. CMES – Computer Modeling in Engineering and Sciences. 2024. vol. 139(3). pp. 3535–3563. DOI: 10.32604/cmes.2023.046658.
18. Zhang Y., Jiang J., Yi C., Li H., Min S., Zuo R., An Z., Yu Y. A Robust CNN for Malware Classification against Executable Adversarial Attack. Electronics. 2024. vol. 13(5). DOI: 10.3390/electronics13050989.
19. Dam T.Q., Nguyen N.T., Le T.V., Le T.D., Uwizeyemungu S., Le-Dinh T. Visualizing Portable Executable Headers for Ransomware Detection: A Deep Learning-Based Approach. Journal of Universal Computer Science. 2024. vol. 30(2). pp. 262–286. DOI: 10.3897/jucs.104901.
20. Gibert D., Zizzo G., Le Q. Towards a Practical Defense Against Adversarial Attacks on Deep Learning-Based Malware Detectors via Randomized Smoothing. Lecture Notes in Computer Science. 2024. vol. 14399. pp. 683–699. DOI: 10.1007/978-3-031-54129-2_40.
21. Zhang P., Wu C., Wang Z. BINCODEX: A comprehensive and multi-level dataset for evaluating binary code similarity detection techniques. BenchCouncil Transactions on Benchmarks, Standards and Evaluations. 2024. vol. 4(2). DOI: 10.1016/j.tbench.2024.100163.
22. Gibert D., Zizzo G., Le Q., Planes J. Adversarial Robustness of Deep Learning-Based Malware Detectors via (De)Randomized Smoothing. IEEE Access. 2024. vol. 12. pp. 61152–61162. DOI: 10.1109/ACCESS.2024.3392391.
23. Louthanova P., Kozak M., Jurecek M., Stamp M., Di Troia F. A comparison of adversarial malware generators. Journal of Computer Virology and Hacking Techniques. 2024. vol. 20. pp. 623–639. DOI: 10.1007/s11416-024-00519-z.
24. Qian L., Cong L. Channel Features and API Frequency-Based Transformer Model for Malware Identification. Sensors. 2024. vol. 24(2). DOI: 10.3390/s24020580.
25. Surendran R., Uddin M.M., Thomas T., Pradeep G. Android Malware Detection Based on Informative Syscall Subsequences. IEEE Access. 2023. vol. 11. DOI: 10.1109/ACCESS.2024.3387475.
26. Kozak M., Jurecek M., Stamp M., Troia F.D. Creating valid adversarial examples of malware. Journal of Computer Virology and Hacking Techniques. 2024. vol. 20. pp. 607–621. DOI: 10.1007/s11416-024-00516-2.
27. Imran M., Appice A., Malerba D. Evaluating Realistic Adversarial Attacks against Machine Learning Models for Windows PE Malware Detection. Future Internet. 2024. vol. 16(5). DOI: 10.3390/fi16050168.
28. Saha S., Afroz S., Rahman A. H. MALIGN: Explainable static raw-byte based malware family classification using sequence alignment. Computers and Security. 2024. vol. 139. DOI: 10.1016/j.cose.2024.103714.
29. Li D., Cui S., Li Y., Xu J., Xiao F., Xu S. PAD: Towards Principled Adversarial Malware Detection Against Evasion Attacks. IEEE Transactions on Dependable and Secure Computing. 2024. vol. 21. no. 2. pp. 920–936. DOI: 10.1109/TDSC.2023.3265665.
30. Zhang F., Li K., Ren Z. Improving Adversarial Robustness of Ensemble Classifiers by Diversified Feature Selection and Stochastic Aggregation. Mathematics. 2024. vol. 12(6). DOI: 10.3390/math12060834.

31. Alzaidy S., Binsalleeh H. Adversarial Attacks with Defense Mechanisms on Convolutional Neural Networks and Recurrent Neural Networks for Malware Classification. Applied Sciences. 2024. vol. 14(4). DOI: 10.3390/app14041673.
32. Zhou K., Wang P., He B. Comparative Study: Mouth Brooding Fish (MBF) as a Novel Approach for Android Malware Detection. International Journal of Advanced Computer Science and Applications. 2024. vol. 15(5). DOI: 10.14569/IJACSA.2024.0150521.
33. Rakib H., Dhakal S.M. Obfuscated Malware Detection: Investigating Real-World Scenarios Through Memory Analysis. In 5th IEEE International Conference on Telecommunications and Photonics (ICTP 2023). 2023. DOI: 10.1109/ICTP60248.2023.10490701.
34. Carrier T., Victor P., Tekeoglu A., Lashkari A.H. Detecting Obfuscated Malware using Memory Feature Engineering. Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP). 2022. vol. 1. pp. 177–188. DOI: 10.5220/0010908200003120.
35. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and prediction (2nd ed.). Springer, 2009. 745 p.
36. Friedman J.H. Greedy function approximation: A gradient boosting machine. Annals of Statistics. 2001. vol. 29(5). pp. 189–1232. DOI: 10.1214/aos/1013203451.
37. Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Ye Q., Liu T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017. pp. 31496–3157. DOI: 10.5555/3294996.3295074.
38. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. vol. 12. pp. 2825–2830. DOI: 10.5555/1953048.2078195.
39. Chen T., Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. pp. 785–794. DOI: 10.1145/2939672.2939785.
40. Lundberg S.M., Lee S.-I. A Unified Approach to Interpreting Model Predictions. 2017. arXiv preprint arXiv:1705.07874. DOI: 10.48550/arXiv.1705.07874.
41. Ribeiro M.T., Singh S., Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. pp. 1135–1144. DOI: 10.1145/2939672.293977.
42. Cevallos-Salas D., Grijalva F., Estrada-Jimenez J., Bentez D., Andrade R. Obfuscated Privacy Malware Classifiers Based on Memory Dumping Analysis. IEEE Access. 2024. vol. 12. pp. 17481–17498. DOI: 10.1109/ACCESS.2024.3358840.
43. Roy K.S., Ahmed T., Udas P.B. Karim M.E., Majumdar S. MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intelligent Systems with Applications. 2023. vol. 20. DOI: 10.1016/j.iswa.2023.200283.

Imamverdiyev Yadigar — Ph.D., Dr.Sci., Head of the department, Cyber security department, Azerbaijan Technical University. Research interests: information security management systems, malware analysis, security of smart systems, security of industrial control systems, web security, cloud security, applied cryptography, biometrics, applications of AI in cyber security. The number of publications — 50. yadigar.imamverdiyev@aztu.edu.az; 25, H. Javid Av., AZ 1073, Baku, Azerbaijan; office phone: +994(50)540-7464.

Baghirov Elshan — Ph.D. candidate, Institute of Information Technology of The Ministry of Science and Education of the Azerbaijan Republic; Senior data scientist, Kapital Bank OJSC. Research interests: malware analysis, data science, information security incident management. The number of publications — 20. elsenbagirov1995@gmail.com; 5/13, A. Kunanbayev St., AZ 1009, Binagadi district, Baku, Azerbaijan; office phone: +994(51)444-1933.

Ikechukwu John Chukwu — Graduate student, Kadir Has University; Ss. Cyril and Methodius University in Skopje (UKIM). Research interests: public-key and lightweight cryptography, quantum optimization problems, malware analysis. The number of publications — 2. cikechukwujohn@stu.khas.edu.tr; Fatih, 34083, Istanbul, Turkey; office phone: +90(212)533-6532.

DOI 10.15622/ia.24.1.5

Я. Имамвердиев, Э. Багиров, Д. Икечукву

ОБНАРУЖЕНИЕ ОБФУСЦИРОВАННЫХ ВРЕДОНОСНЫХ ПРОГРАММ В WINDOWS С ПОМОЩЬЮ МЕТОДОВ АНСАМБЛЕВОГО ОБУЧЕНИЯ

Имамвердиев Я., Багиров Э., Икечукву Д. **Обнаружение обфусцированных вредоносных программ в Windows с помощью методов ансамблевого обучения.**

Аннотация. В эпоху Интернета и смарт-устройств обнаружение вредоносных программ стало важным фактором для безопасности системы. Обфусцированные вредоносные программы создают значительные риски для различных платформ, включая компьютеры, мобильные устройства и устройства IoT, поскольку не позволяют использовать передовые решения для обеспечения безопасности. Традиционные эвристические и сигнатурные методы часто не справляются с этими угрозами. Поэтому была предложена экономически эффективная система обнаружения с использованием анализа дампа памяти и методов ансамблевого обучения. На основе набора данных CIC-MalMem-2022 была оценена эффективность деревьев решений, градиентного бустинга деревьев, логистической регрессии, метода случайного леса и LightGBM при выявлении обфусцированных вредоносных программ. Исследование продемонстрировало превосходство методов ансамблевого обучения в повышении точности и надежности обнаружения. Кроме того, SHAP (аддитивные объяснения Шелли) и LIME (локально интерпретируемые объяснения, не зависящие от устройства модели) использовались для выяснения прогнозов модели, повышения прозрачности и надежности. Анализ выявил важные особенности, существенно влияющие на обнаружение вредоносных программ, такие как службы процессов, активные службы, дескрипторы файлов, ключи реестра и функции обратного вызова. Эти идеи имеют большое значение для совершенствования стратегий обнаружения и повышения производительности модели. Полученные результаты вносят вклад в усилия по обеспечению кибербезопасности путем всесторонней оценки алгоритмов машинного обучения для обнаружения обфусцированных вредоносных программ с помощью анализа памяти. В этой статье представлены ценные идеи для будущих исследований и достижений в области обнаружения вредоносных программ, прокладывая путь для более надежных и эффективных решений в области кибербезопасности перед лицом развивающихся и сложных вредоносных угроз.

Ключевые слова: обнаружение вредоносного ПО, машинное обучение, анализ вредоносного ПО, кибербезопасность.

Литература

1. Baghirov E. Evaluating the performance of different machine learning algorithms for Android malware detection. In 2023 5th International Conference on Problems of Cybernetics and Informatics (PCI). IEEE, 2023. pp. 1–4. DOI: 10.1109/PCI60110.2023.10326006.
2. Baghirov E. Comprehensive framework for malware detection: Using ensemble methods, feature selection, and hyperparameter optimization. In 2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT). IEEE, 2023. pp. 1–5. DOI: 10.1109/AICT59525.2023.10313179.

3. Jeon J., Jeong B., Baek S., Jeong Y.-S. Static Multi Feature-Based Malware Detection Using Multi SPP-net in Smart IoT Environments. IEEE Transactions on Information Forensics and Security. 2024. vol. 19. pp. 2487–2500. DOI: 10.1109/TIFS.2024.3350379.
4. Ismail S.J.I., Hendrawan Rahardjo B., Juhana T., Musashi Y. MalSSL – Self-Supervised Learning for Accurate and Label-Efficient Malware Classification. IEEE Access. 2024. vol. 12. pp. 58823–58835. DOI: 10.1109/ACCESS.2024.3392251.
5. Baghirov E. Malware detection based on opcode frequency. Journal of Problems of Information Technology, 2023. vol. 14(1). pp. 3–7. DOI: 10.25045/jpit.v14.i1.01.
6. Egitmen A., Yavuz A.G., Yavuz S. TRConv: Multi-Platform Malware Classification via Target Regulated Convolutions. IEEE Access. 2024. vol. 12. pp. 71492–71504. DOI: 10.1109/ACCESS.2024.3401627.
7. Gungor A., Dogru I.A., Barisci N., Toklu S. Malware detection using image-based features and machine learning methods. Journal of the Faculty of Engineering and Architecture of Gazi University. 2023. vol. 38. no. 3. pp. 1781–1792. DOI: 10.17341/gazimmfd.994289.
8. Mesbah A., Baddari I., Riahla M.A. LongCGDroid: Android malware detection through longitudinal study for machine learning and deep learning. Jordanian Journal of Computers and Information Technology. 2023. vol. 9. no. 4. pp. 328–346. DOI: 10.5455/jjcit.71-1693392249.
9. Howard A., Hope B., Saltaformaggio B., Avena E., Ahmadi M., Duncan M., McCann R., Cukierski W. Microsoft Malware Prediction. Kaggle, 2018. Available at: https://kaggle.com/competitions/microsoft-malware-prediction. (accessed 26.10.2024).
10. Ahmed I.T., Hammad B.T., Jamil N.A Comparative Performance Analysis of Malware Detection Algorithms Based on Various Texture Features and Classifiers. IEEE Access. 2024. vol. 12. pp. 11500–11519. DOI: 10.1109/ACCESS.2024.3354959.
11. Xie W., Zhang X. The Application of Machine Learning in Android Malware Detection. 2024 4th International Conference on Neural Networks, Information and Communication Engineering (NNICE). 2024. pp. 1–4. DOI: 10.1109/NNICE61279.2024.10498936.
12. Bostani H.; Moonsamy V. EvadeDroid: A practical evasion attack on machine learning for black-box Android malware detection. Computers and Security. 2024. vol. 139. DOI: 10.1016/j.cose.2023.103676.
13. Rigaki M., Garcia S. The Power of MEME: Adversarial Malware Creation with Model-Based Reinforcement Learning. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2024. pp. 44–64. DOI: 10.1007/978-3-031-51482-1 _3.
14. Rudd E.M., Krisiloff D., Coull S., Olszewski D., Raff E., Holt J. Efficient Malware Analysis Using Metric Embeddings. Digital Threats: Research and Practice. 2024. vol. 5(1). pp. 1–20. DOI: 10.1145/3615669.
15. Zhan D., Zhang Y., Zhu L., Chen J., Xia S., Guo S., Pan Z. Enhancing reinforcement learning based adversarial malware generation to evade static detection. Alexandria Engineering Journal. 2024. vol. 98. pp. 32–43. DOI: 10.1016/j.aej.2024.04.024.
16. Aljabri M., Alhaidari F., Albuainain A., Alrashidi S., Alansari J., Alqahtani W., Alshaya J. Ransomware detection based on machine learning using memory features. Egyptian Informatics Journal. 2024. vol. 25. DOI: 10.1016/j.eij.2024.100445.
17. Ban Y., Kim M., Cho H. An Empirical Study on the Effectiveness of Adversarial Examples in Malware Detection. CMES – Computer Modeling in Engineering and Sciences. 2024. vol. 139(3). pp. 3535–3563. DOI: 10.32604/cmes.2023.046658.

18. Zhang Y., Jiang J., Yi C., Li H., Min S., Zuo R., An Z., Yu Y. A Robust CNN for Malware Classification against Executable Adversarial Attack. Electronics. 2024. vol. 13(5). DOI: 10.3390/electronics13050989.
19. Dam T.Q., Nguyen N.T., Le T.V., Le T.D., Uwizeyemungu S., Le-Dinh T. Visualizing Portable Executable Headers for Ransomware Detection: A Deep Learning-Based Approach. Journal of Universal Computer Science. 2024. vol. 30(2). pp. 262–286. DOI: 10.3897/jucs.104901.
20. Gibert D., Zizzo G., Le Q. Towards a Practical Defense Against Adversarial Attacks on Deep Learning-Based Malware Detectors via Randomized Smoothing. Lecture Notes in Computer Science. 2024. vol. 14399. pp. 683–699. DOI: 10.1007/978-3-031-54129-2_40.
21. Zhang P., Wu C., Wang Z. BINCODEX: A comprehensive and multi-level dataset for evaluating binary code similarity detection techniques. BenchCouncil Transactions on Benchmarks, Standards and Evaluations. 2024. vol. 4(2). DOI: 10.1016/j.tbench.2024.100163.
22. Gibert D., Zizzo G., Le Q., Planes J. Adversarial Robustness of Deep Learning-Based Malware Detectors via (De)Randomized Smoothing. IEEE Access. 2024. vol. 12. pp. 61152–61162. DOI: 10.1109/ACCESS.2024.3392391.
23. Louthanova P., Kozak M., Jurecek M., Stamp M., Di Troia F. A comparison of adversarial malware generators. Journal of Computer Virology and Hacking Techniques. 2024. vol. 20. pp. 623–639. DOI: 10.1007/s11416-024-00519-z.
24. Qian L., Cong L. Channel Features and API Frequency-Based Transformer Model for Malware Identification. Sensors. 2024. vol. 24(2). DOI: 10.3390/s24020580.
25. Surendran R., Uddin M.M., Thomas T., Pradeep G. Android Malware Detection Based on Informative Syscall Subsequences. IEEE Access. 2023. vol. 11. DOI: 10.1109/ACCESS.2024.3387475.
26. Kozak M., Jurecek M., Stamp M., Troia F.D. Creating valid adversarial examples of malware. Journal of Computer Virology and Hacking Techniques. 2024. vol. 20. pp. 607–621. DOI: 10.1007/s11416-024-00516-2.
27. Imran M., Appice A., Malerba D. Evaluating Realistic Adversarial Attacks against Machine Learning Models for Windows PE Malware Detection. Future Internet. 2024. vol. 16(5). DOI: 10.3390/fi16050168.
28. Saha S., Afroz S., Rahman A. H. MALIGN: Explainable static raw-byte based malware family classification using sequence alignment. Computers and Security. 2024. vol. 139. DOI: 10.1016/j.cose.2024.103714.
29. Li D., Cui S., Li Y., Xu J., Xiao F., Xu S. PAD: Towards Principled Adversarial Malware Detection Against Evasion Attacks. IEEE Transactions on Dependable and Secure Computing. 2024. vol. 21. no. 2. pp. 920–936. DOI: 10.1109/TDSC.2023.3265665.
30. Zhang F., Li K., Ren Z. Improving Adversarial Robustness of Ensemble Classifiers by Diversified Feature Selection and Stochastic Aggregation. Mathematics. 2024. vol. 12(6). DOI: 10.3390/math12060834.
31. Alzaidy S., Binsalleeh H. Adversarial Attacks with Defense Mechanisms on Convolutional Neural Networks and Recurrent Neural Networks for Malware Classification. Applied Sciences. 2024. vol. 14(4). DOI: 10.3390/app14041673.
32. Zhou K., Wang P., He B. Comparative Study: Mouth Brooding Fish (MBF) as a Novel Approach for Android Malware Detection. International Journal of Advanced Computer Science and Applications. 2024. vol. 15(5). DOI: 10.14569/IJACSA.2024.0150521.
33. Rakib H., Dhakal S.M. Obfuscated Malware Detection: Investigating Real-World Scenarios Through Memory Analysis. In 5th IEEE International

Conference on Telecommunications and Photonics (ICTP 2023). 2023. DOI: 10.1109/ICTP60248.2023.10490701.

34. Carrier T., Victor P., Tekeoglu A., Lashkari A.H. Detecting Obfuscated Malware using Memory Feature Engineering. Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP). 2022. vol. 1. pp. 177–188. DOI: 10.5220/0010908200003120.
35. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and prediction (2nd ed.). Springer, 2009. 745 p.
36. Friedman J.H. Greedy function approximation: A gradient boosting machine. Annals of Statistics. 2001. vol. 29(5). pp. 189–1232. DOI: 10.1214/aos/1013203451.
37. Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Ye Q., Liu T. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017. pp. 31496–3157. DOI: 10.5555/3294996.3295074.
38. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. vol. 12. pp. 2825–2830. DOI: 10.5555/1953048.2078195.
39. Chen T., Guestrin C. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. pp. 785–794. DOI: 10.1145/2939672.2939785.
40. Lundberg S.M., Lee S.-I. A Unified Approach to Interpreting Model Predictions. 2017. arXiv preprint arXiv:1705.07874. DOI: 10.48550/arXiv.1705.07874.
41. Ribeiro M.T., Singh S., Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016. pp. 1135–1144. DOI: 10.1145/2939672.293977.
42. Cevallos-Salas D., Grijalva F., Estrada-Jimenez J., Bentez D., Andrade R. Obfuscated Privacy Malware Classifiers Based on Memory Dumping Analysis. IEEE Access. 2024. vol. 12. pp. 17481–17498. DOI: 10.1109/ACCESS.2024.3358840.
43. Roy K.S., Ahmed T., Udas P.B. Karim M.E., Majumdar S. MalHyStack: A hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis. Intelligent Systems with Applications. 2023. vol. 20. DOI: 10.1016/j.iswa.2023.200283.

Имамвердиев Ядигар — Ph.D., Dr.Sci., заведующий кафедрой, заведующий кафедрой кибербезопасности, Азербайджанский технический университет. Область научных интересов: системы управления информационной безопасностью, анализ вредоносных программ, безопасность интеллектуальных систем, безопасность промышленных систем управления, веб-безопасность, облачная безопасность, прикладная криптографию, биометрия, применение искусственного интеллекта в кибербезопасности. Число научных публикаций — 50. yadigar.imamverdiyev@aztu.edu.az; проспект Х. Джавида, 25, AZ 1073, Баку, Азербайджан; р.т.: +994(50)540-7464.

Багиров Эльшан — аспирант, Институт информационных технологий Министерства науки и образования Азербайджанской Республики; старший специалист по обработке данных, ОАО "Капитал Банк". Область научных интересов: анализ вредоносных программ, обработка данных, управление инцидентами информационной безопасности. Число научных публикаций — 20. elsenbagirov1995@gmail.com; улица А. Кунанбаева, 5/13, AZ 1009, Бинагадинский район, Баку, Азербайджан; р.т.: +994(51)444-1933.

Икечукву Джон Чукву — аспирант, Университет Кадир Хас; Университет святых Кирилла и Мефодия в Скопье (UKIM). Область научных интересов: криптография с открытым ключом и облегченная криптография, проблемы квантовой оптимизации, анализ вредоносных программ. Число научных публикаций — 2. cikechukwujohn@stu.khas.edu.tr; Фатих, 34083, Стамбул, Турция; р.т.: +90(212)533-6532.