

A Hybrid Anomaly Detection Approach for Obfuscated Malware

Gerard Shu Fuhnwi

*Gianforte School of Computing
Montana State University
Bozeman, MT, USA*

gerard.shufuhnwi@student.montana.edu

Matthew Revelle

*Gianforte School of Computing
Montana State University
Bozeman, MT, USA*

matthew.revelle@montana.edu

Clemente Izurieta

*Gianforte School of Computing
Montana State University
Pacific Northwest National Laboratory
Idaho National Laboratory
Bozeman, MT, USA*

clemente.izurieta@montana.edu

Abstract—With the rapid evolution of malicious software, cyber threats have become increasingly sophisticated, employing advanced obfuscation techniques to evade traditional detection methods. This study presents a hybrid anomaly detection approach applied to obfuscated malware. Even though there is a large body of research in this field, existing malware detection techniques have drawbacks, such as requiring large amounts of data, trustworthiness (imprecise results) of algorithms, and advanced obfuscation. There is a need to employ solid and efficient techniques for malware detection to overcome these challenges. This paper proposes a hybrid approach, combining an autoencoder with traditional machine-learning methods to create an efficient malware detection framework. We used the malware memory dataset (MalMemAnalysis-2022) to evaluate this framework. The experimental results show our proposed approach can detect obfuscated malware when a deep autoencoder used for feature learning is combined with logistic regression. It is extremely fast with an Accuracy, Detection Rate (DR), Matthew Correlation Coefficient(MCC), and Statistical Parity Difference (SPD) of 99.97%, 99.98%, 99.93%, and 0.03%, respectively.

Index Terms—Malware Detection, Hybrid Anomaly Detection, Obfuscated Malware, Deep Autoencoder, Logistic Regression

1. Introduction

In an era dominated by interconnected technologies and digital dependence, the cybersecurity landscape faces an ever-evolving and persistent threat—malware. Malware are malicious programs that can potentially compromise computer systems' integrity, confidentiality, and availability. As a pervasive threat, malware manifests itself in diverse forms, each using different tactics, techniques, and procedures (TTPs). Malware can result in consequences ranging from website defacement [1] to the loss of human life [2]. Moreover, the constant evolution of defense evasion techniques in malware continues to require advancement in detection methods.

There are many categories of malware, such as viruses, trojan horses, spyware, ransomware, bots, bot-

nets, worms, and others [3]. While there are many malware categories, there are a number of common tactics and techniques [3] shared between different types of malware. A detection system that utilizes features which indicate the implementation of those tactic and techniques will be capable of detecting malware of various categories. To counteract malware infiltrating systems, gaining access to information, accessing authorized users, and other cybercrimes, the best alternative is to create detection systems capable of detecting and stopping all forms of malware.

Malware is frequently obfuscated to evade signature-based detection methods [4], and to increase the effort required to reverse engineer the malware when it has been collected. Obfuscation can hide the implementation of malicious behaviors from static analysis and require sophisticated detection techniques [5]. These detection methods illustrate the applications of anomaly detection in Machine Learning (ML), which involves identifying data patterns (outcomes, values, or observations) that differ significantly from the rest of the data and are essential for identifying and preventing malware [5]. The basic intuition of these learning systems is figuring out which training data to fit into the machine learning method to make the quickest and most accurate assessment. Anomaly detection techniques perform malware detection by attempting to identify malicious behavior. These anomaly detection techniques use their knowledge of what constitutes normal behavior to decide the maliciousness of a program under inspection. Anomaly detection systems take in a set of features together with large sample sizes to compare and contrast differences. These features can be input in different formats, which is a factor that determines how the anomaly detection system should be used. While some anomaly detection algorithms are focused on minimizing false positives, others are focused on speed, scalability, accuracy, and precision. Therefore, choosing different algorithms corresponding to the objective and input type has an enormous impact on the result of the anomaly detection system. Hybrid anomaly detection is one such technique, which combines two or more ML models to enhance the

overall effectiveness of anomaly detection.

Several approaches to malware detection using hybrid anomaly detection techniques [6] exist. However, in most of the work, complexity, and time consumption are high, fairness of the algorithms are not mentioned and there is no statistical analysis to compare the various models, making them unsuitable for real-world applications. This motivates the development of a fair, fast, efficient, and easy-to-interpret method for obfuscated malware detection using the most relevant features captured through autoencoders.

Main Contributions: The main contributions of this research include:

- An anomaly detection approach for malware analysis, combining a deep autoencoder and logistic regression to increase trustworthiness and transparency in current obfuscated malware detection. To the best of our knowledge, this is the first time this approach has been used in anomaly detection.
- Analysis of the performance of our approach. We perform a comparative analysis of combining a deep autoencoder with logistic regression against state-of-the-art and baseline methods.
- Improvements to the overall anomaly detection speed, fairness, and reduced ML model complexity.
- Improvements in accuracy, detection rates, and MCC when compared to other models and benchmarks in related works with very low statistical parity difference (SPD).

The rest of this paper is organized as follows. Section II discusses related work. Section III proposes a hybrid anomaly detection framework for obfuscated malware including dataset description, data transformation, deep autoencoder, and logistic regression. In Section IV we analyze empirical results, and Section V concludes the paper.

2. Related Work

Many studies have been conducted to protect devices connected to the Internet from malware intrusion using anomaly detection-based techniques. Research on static anomaly detection, which identifies the program's file structure characteristics under inspection, is used to detect malicious code [6-7]. However, static anomaly detection techniques can not detect obfuscated or hidden malware because of their inability to execute software. Conversely, dynamic anomaly detection techniques can detect malicious code using information gathered from the program's execution, making them capable of detecting obfuscated or hidden malware [8-14]. On the other hand, hybrid anomaly detection techniques can detect malware by combining both static and dynamic anomaly detection techniques [3], [16], [17].

This section highlights previous studies on anomaly detection techniques for malware detection through static, dynamic, and hybrid methods.

2.1. Static Anomaly Detection

Li et al. [7] proposed a Fileprint (n-gram) analysis as a means to detect malware. During the training phase, an ML model or set of models is derived to characterize various file types of a system based on their structural (byte) composition. Li et al.'s technique exhibited 94.5% detection rate for PDF files with embedded malware. Zhang et al. [8] proposed a deep learning-based malware detection framework for classifying eight types of ransomware. To reduce the dimensionality of the opcode sequences used as features, the term frequency-inverse document frequency (TF-IDF) algorithm was applied to filter out N-gram Opcodes with low amount of information according to their TF-IDF values. After the opcode sequence was divided into several patches, a self-attention convolutional neural network (SA-CNN) model was trained using these patches. Finally, to analyze the relationship between opcode sequences and classify the ransomware, a directional self-attention network (Di-SAN) model was used.

2.2. Dynamic Anomaly Detection

Wang and Stolfo [9] proposed a tool called PAYL, which calculates the expected payload for each service (port) on a system by using a centroid model. The detector compares incoming payloads with the centroid model by measuring the Mahalanobis distance [27] between the two. The author's technique had an overall detection rate of approximately 60 % and a false positive rate of 1 % or lower.

Lee and Stolfo [10] proposed using data mining techniques, association rules, and frequent episodes for intrusion detection systems. The association rules and frequent episodes are collectively referred to as rule sets and serve as knowledge of what is normal for the host's services.

Boldt and Carlson [11] proposed a Forensic Tool Kit (FTK) to identify Privacy-Invasive Software (PIS) such as adware and spyware. The basic approach for the baseline consists of initially creating a system free of PIS, that is, a clean system. This tool is used to depict the file system of the target host. Boldt and Carlson discovered that their technique produced false positives and negatives for Ad-Aware.

Hofmeyr et al. [12] proposed a technique to help monitor system call sequences to detect maliciousness by developing profiles that represent the normal behavior of the system's services. The Hamming distance determines how closely the system call sequence resembles another, where processes with large Hamming distances are considered abnormal. The Hofmeyr et al. technique found intrusions that attempted to exploit

various UNIX programs like sendmail, lpr, and ftpd. Sekar et al. [13] proposed a Finite State Automata (FSA) for anomaly detection. Each node in the FSA represents a state (program counter) in the program (static)/process (dynamic) under inspection (PUI), which the algorithm utilizes to learn normal data faster and perform better detection, where system calls give transitions in the FSA. When a system call is invoked, a new transition is added to the automation, where automations resulting from multiple executions are considered normal. Sato et al. [14] proposed an n-gram approach and compared it to the FSA approach [13]. Sato et al. evaluated the two approaches on *htpd*, *ftpd*, and *nsfd*, where the FSA was found to have a lower false positive rate when compared to the n-gram approach. Joen et al. [15] proposed a decreasing additive-increase/multiplicative-decrease (DAIMD) model for detecting new and variant IoT malware. Performing dynamic analysis in a closed-based nested virtual environment, network, process, memory, and virtual file system behaviors were extracted as features and converted into images. These images are used to train a CNN model for detecting IoT malware.

2.3. Hybrid Anomaly Detection

Carrier et al. [3] proposed using VolMemLyzer to detect obfuscated malware. Using a memory feature extraction technique called VolMemLyzer, base learners and meta-learners are combined to detect obfuscated malware by stacking them together. By combining naive Bayes, random forest, and decision trees as base learners with logistic regression as the meta-learner, the authors obtained an accuracy of 99.00% and an F_1 score of

99.02%. Wang et al. [16] proposed a technique called the cross-view difference approach for detecting a type of malware known as ghostware, which is malware that attempts to hide its existence from the operating system's querying utilities. For example, the proposed approach to mitigate hidden files requires comparing results from a user-mode program like *dir* or corresponding API calls to the equivalent data found by inspecting the file system data structures directly.

Darabian et al. [17] proposed a method for detecting cryptomining malware through hybrid analysis. Using opcodes extracted from static analysis and system calls generated from dynamic analysis, then LSTM, attention-based LSTM (ATT-LSTM), and CNN are used to evaluate their performance for classification by a softmax function.

Overall, these studies demonstrate the effectiveness of anomaly detection techniques and feature extraction for malware detection. Our study builds on these papers by extending the work of Tristan et al. [3] on obfuscated malware detection, and mitigates the limitations mentioned above by combining a deep autoencoder with logistic regression on the obfuscated malware dataset. We demonstrate high accuracy, MCC, and detection rates in less computational time. Our proposed approach also has a low statistical parity difference, indicating how fair, trustworthy, and transparent our approach is.

3. Proposed Model Approach

We propose a hybrid anomaly detection approach for malware analysis. The step-by-step process adopted in

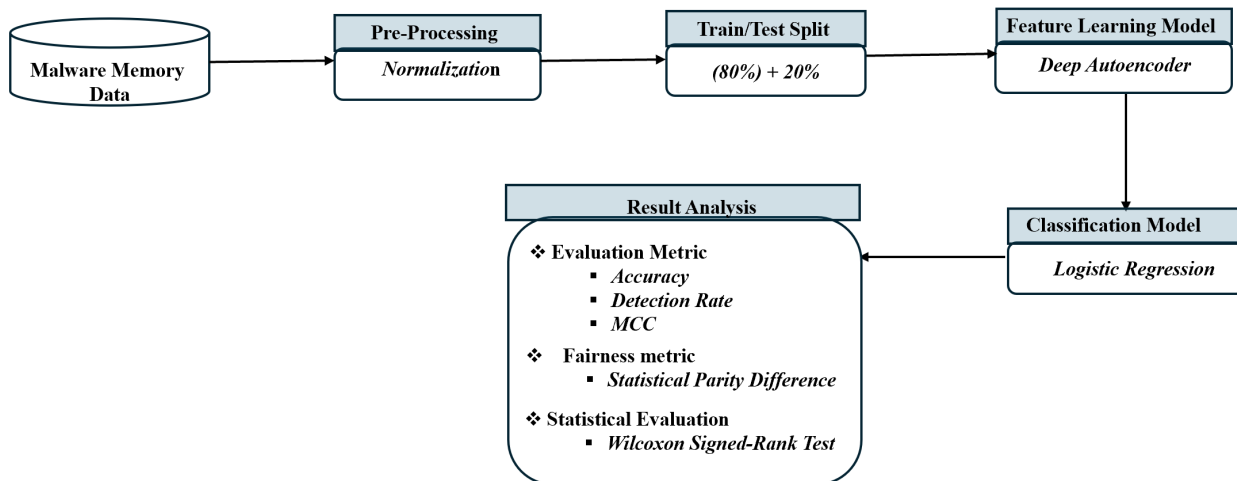


Figure 1. Pipeline of our hybrid approach. Rectangles represent the model's processes, cylinders represent stored data, and the arrows the direction of flow

the proposed methodology is described herein and shown in Figure 1.

3.1. Malware Memory Data

The malware memory dataset is balanced, comprising 50% malicious and 50% benign memory dumps. Malicious dumps include different malware categories: Ransomware, Spyware, and Trojan Horse. The dataset contains 58,596 records, with 29,298 being benign, 29,298 being malicious, and 55 features. This dataset is designed to test obfuscated malware detection methods through memory [3].

3.2. Preprocessing

The main objective of data preprocessing is to enhance the quality of the data, making it easier to work with and enabling more accurate and efficient results for ML algorithms [18].

- **Normalization:** Normalization is a technique that scales numerical variables in a particular range, typically [0, 1] or [-1, 1]. Normalization aims to bring different features into a similar scale, making it easier for ML algorithms to process the data and reduce the risk of certain features dominating the model due to their larger numerical values. Several normalization methods; the most common are Min-Max Scaling and Z-score.
- **Standardization:** In our proposed approach, the Z-score standardization, which scales the value of the entire feature to a standard range of [0,1], is used. In Z-score standardization, the average μ of each numerical feature is calculated and then subtracted from the feature value x . The subtracted average and feature x are then divided by the standard deviation σ as shown in the formula:

$$X_i = \frac{x - \mu}{\sigma}$$

3.3. Train/Test Split

The 80/20 split in ML refers to dividing your dataset into two portions: an 80% portion for training your model and a 20% portion for testing its performance. This division is a common practice to assess how well your model generalizes to new, unseen data. The 80/20 split is a common choice because it balances having enough training data and a sufficiently large test set to evaluate performance.

3.4. Feature Learning Model

A feature learning model is a cornerstone in machine learning, designed not only to automatically extract

relevant features or representations from raw data but also to uphold principles of fairness. Instead of relying solely on manually engineered features, these models can autonomously unearth hierarchical and meaningful representations from the input data while ensuring fairness across different demographic groups. One example of a feature learning model that integrates fairness considerations is a deep autoencoder, exemplifying the power of learning intricate and informative representations through neural network architectures while promoting equitable outcomes in decision-making processes.

3.4.1. Deep Autoencoder. A deep autoencoder, a type of neural network architecture, is a key player in the realm of unsupervised learning. It's comprised of two neural networks: an encoder and a decoder. The deep autoencoder's primary objective is to acquire a compact and informative input data representation. The encoder efficiently compresses the input into a lower-dimensional representation (encoding), and the decoder skillfully reconstructs the original input from this representation [28]. The architecture of a deep autoencoder is as follows:

- **Encoder:** The encoder network transforms the raw input data into a lower-dimensional representation. Each layer of the encoder extracts increasingly abstract and higher-level features.
- **Bottleneck Layer:** This is the layer in the middle of the network where the input is highly compressed. It contains the learned features or encoding of the input data.
- **Decoder:** The decoder network takes the compressed representation from the bottleneck layer and reconstructs the original input data. Like the encoder, each layer of the decoder progressively refines the reconstruction.

3.5. Classification Model

For malware analysis, we used logistic regression, a supervised learning algorithm, to classify obfuscated malware into benign and malicious memory dumps.

- **Logistic Regression:** Logistic regression is a widely used algorithm because of its versatility and interpretability. It is beneficial when binary classification, probability estimation, and simplicity are essential. Logistic regression is a statistical technique that helps predict the probability of obfuscated malware belonging to benign and malicious memory dumps.

3.6. Result Analysis

The result analysis includes evaluation metrics and assessment of their statistical significance.

3.6.1. Evaluation and Fairness Metrics. Once the logistic regression classifier is trained using the 80% portion of the dataset, it is evaluated using the 20% portion. The classifier’s performance is evaluated using accuracy, detection rate (DR), and Matthews Correlation Coefficient (MCC), while fairness is assessed using the statistical parity difference (SPD). These performance metrics are calculated using True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN) as shown in the confusion matrix [19] in Table 1.

TABLE 1. CONFUSION MATRIX: A CONTINGENCY CONTAINING FOUR METRICS, TRUE POSITIVE (TP), TRUE NEGATIVE (TN), FALSE POSITIVE (FP), AND FALSE NEGATIVE (FN).

Memory Dumps		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

- **Accuracy:** Is defined as the total number of correctly classified memory dumps divided by the total number of memory dumps which can be calculated using the formula:

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

- **Detection Rate:** It is the ratio between the total number of malicious memory dumps detected by the logistic regression classifier to the total number of malicious memory dumps present in the dataset [20] which can be calculated using the formula:

$$DR = \frac{TP}{TP + FN}$$

- **Matthews Correlation Coefficient (MCC):** Is defined as the Pearson product-moment correlation coefficient using the confusion matrix in Table 1 between the actual malicious memory dumps and predicted malicious memory dumps [21] which can be calculated using the formula below.

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}}$$

- **Statistical Parity Difference (SPD):** SPD measures the difference in the proportion of malicious memory dumps between different memory dumps. It aims to ensure the model’s predictions are balanced across the memory dumps, indicating fairness and lack of bias in its decision-making process. Smaller values indicate a disparity in the proportion of malicious predictions between the memory dumps, which is calculated using the formula:

$$SPD = \frac{TP}{TP + FN} - \frac{TN}{TN + FP}$$

3.6.2. Statistical Evaluation. We utilize the Wilcoxon signed-rank test to assess the statistical significance of our proposed approach. This non-parametric statistical hypothesis test ranks the differences between two classifiers, disregards the signs, and compares the ranks of the positive and negative differences [22]. It is often used when the assumptions of a paired t-test are violated (such as when the data is not normally distributed). To compare the performance of our proposed approach against logistic regression, we used the Wilcoxon signed-rank test to test whether there is a significant difference between the performances on the median of accuracy, detection rate, and MCC. The null hypothesis (H_0) for the Wilcoxon signed-rank test states that there is no significant difference between the median performances of the two models. In contrast, the alternative hypothesis (H_1) states a statistically significant difference, unlikely to have occurred by chance, between the median performances of the two models.

4. Experimental Results

All experiments were performed in Python using default parameters for logistic regression ¹. The deep autoencoder consisted of two encoder levels, a bottleneck, and two decoder levels. All experiments were run on Intel (R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30GHz processor with 16 GB RAM. Logistic regression Training and testing were performed on the 55 features and logistic regression with the features learned from the autoencoder with 100 different random seed values. An 80%/20% training/testing split of the data was used. The proposed approach was evaluated by finding the median value of the accuracy, detection rate, and MCC metrics. The highlighted row in Table 2 shows the performance of our proposed approach. It is evident from Table 2 that our proposed approach has high accuracy, detection rate, and MCC.

Our proposed approach has a low SPD, which ensures that the model’s predictions are fair and equitable across the different memory dumps, promoting trust and transparency.

The Wilcoxon signed-rank test was used to compare the differences between our proposed approach and logistic regression. Table 3 compares our proposed approach against logistic regression on all metrics used. From Table 3, comparing our proposed approach against logistic regression, we can see that the p-values of the test for accuracy, detection rate, and MCC are all significant at $\alpha = 0.05$. We can statistically confirm that our proposed approach outperforms logistic regression.

The highlighted last column in Table 2 shows the approximate run times for a memory dump to be classified as malicious or benign for all samples in the 20%

¹Sklearn

TABLE 2. ACCURACY, DETECTION RATE, MCC, SPD, AND CLASSIFICATION TIME. THE BEST RESULTS ARE PRINTED IN SKYBLUE.

Model	Accuracy	Detection Rate	MCC	SPD	Classification Time (seconds)
Logistic Regression	0.9962	0.9966	0.9923	0.0009	1.2665
Proposed Approach	0.9997	0.9998	0.9993	0.0003	0.8541

TABLE 3. RESULTS OF WILCOXON SIGNED-RANK TEST COMPARING OUR PROPOSED APPROACH AGAINST LOGISTIC REGRESSION

Comparison	Evaluation metric	Positive Rank	Negative Rank	Rank (Test Statistics)	Hypothesis ($\alpha = 0.05$)	p-value
Proposed Approach vs. Logistic Regression with significant values highlighted in table 2	Accuracy	5050	0	5050	Rejected	<0.001
	Detection Rate	5050	0	5050	Rejected	< 0.001
	MCC	5050	0	5050	Rejected	< 0.001

TABLE 4. COMPARISON WITH OTHER OBFUSCATED MALWARE DETECTION METHODS

Method	Accuracy	Detection Rate	MCC	Speed	Statistical Test	Complexity	SPD
Tristan et al. [3]	High	Not Mentioned	Not Mentioned	Very High	Not Mentioned	Medium	Not Mentioned
Xu et al. [23]	Very High	Not Mentioned	Not Mentioned	Medium	Not Mentioned	Medium	Not Mentioned
Bozkir et al. [24]	High	Not Mentioned	Not Mentioned	Medium	Not Mentioned	High	Not Mentioned
Nissima et al. [25]	High	Not Mentioned	Not Mentioned	Medium	Not Mentioned	Medium	Not Mentioned
Javaheri and Hosseinizadeh [26]	Medium	Not Mentioned	Not Mentioned	Medium	Not Mentioned	High	Not Mentioned
Proposed Approach	Very High	Very High	Very High	Extremely High	Included	Low	Included

test data. From the table, we can see that the classification time is linear, which demonstrates our proposed approach’s scalability. Our proposed approach has a classification time of 0.00007 milliseconds per sample, significantly lower than all models listed in Table Table 4. Comparing these related works, as shown in Table 4, shows the relevance of our proposed approach as a fast method for detecting obfuscated or hidden malware.

5. Conclusion and Future work

In this paper, a hybrid anomaly detection method combining a deep autoencoder and logistic regression was implemented on an obfuscated malware dataset. We compared our proposed approach with logistic regression using the following metrics: accuracy, detection rate, MCC, and SPD. Because of feature learning, the

computational cost is decreased, and our proposed approach has an accuracy of 99.97%, a detection rate of 99.98%, an MCC of 99.93%, and an SPD of 0.03%. We also compared our proposed approach with logistic regression using the Wilcoxon signed-rank test and found that our proposed approach is better or competitive than state-of-the-art and baseline methods. Moreover, our proposed approach is compared with related works on malware detection. The comparison results in Table 4 showed that a deep autoencoder with logistic regression has better performance, low SPD, and requires less classification time.

For future work, we would like to use model interpretation tools such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) to gain insights into potential sources of bias.

References

- [1] Adiga, Narasimha R and Almási, George and Almasi, George S and Aridor, Yariv and Barik, Rajkishore and Beece, D and Bellofatto, Ralph and Bhanot, Gyan and Bickford, Randy and Blumrich, M and others, "An overview of the BlueGene/L supercomputer," SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, pp. 60–60, 2002.
- [2] CARNEGIE-MELLON UNIV PITTSBURGH PA, CERT/CC 2003 Annual Report, May 2003.
- [3] Carrier, Tristan, "Detecting obfuscated malware using memory feature engineering," University of New Brunswick, 2022.
- [4] Bahador MB, Abadi M, and Tajoddin A, "LMD: a signature-based approach to hardware-level behavioral malware detection and classification," *The Journal of Supercomputing*, Springer, pp.5551–5582, (2019).
- [5] Fuhnwi, Gerard Shu and Agbaje, Janet O and Oshinubi, Kayode and Peter, Olumuyiwa James: An Empirical Study on Anomaly Detection Using Density-based and Representative-based Clustering Algorithms, *Journal of the Nigerian Society of Physical Sciences*, pp.1364-1364, (2023).
- [6] Idika, Nwokedi and Mathur, Aditya P, "A survey of malware detection techniques," *Purdue University*, vol.48, no.2, pp.32–46, 2007.
- [7] W.Li, K.Wang, S.Stolfo, and B.Herzog, "Fileprints: Identifying file types by n-gram analysis," *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, IEEE, pp.64–71, 2005.
- [8] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes," *Future Generation Computer Systems*, Elsevier, vol. 110, pp. 708–720, 2020.
- [9] K.Wang and S.J.Stolfo, "Anomalous payload-based network intrusion detection," In *Proceedings of Recent Advance in Intrusion Detection (RAID)*, pp. 201–222, September 2004.
- [10] W.Lee and S.Stolfo, "Anomalous payload-based network intrusion detection," In *Proceedings of 7th USENIX Security Symposium*, 1998.
- [11] M.Boldt and B.Carlsson, "Analysis privacy-invasive software using computer forensic methods," January 2006.
- [12] S.Hofmeyr, S.Forrest, and A.Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, pp. 151–180, 1998.
- [13] R.Sekar, M.Bendre, P.Bollineni, and D.Dhurjati, "A fast automaton-based approach for detecting anomalous program behaviors," In *IEEE Symposium on Security and Privacy*, 2001.
- [14] I.Sato, Y.Okazaki, and S.Goto, "An improved intrusion detection method based on process profiling," *IPSJ Journal*, vol. 43, no. 11, pp. 3316–3326, 2002.
- [15] J. Jeon, J. H. Park, and Y. S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, 2020.
- [16] Y.M.Wang, D.Beck, B.Vo, R.Roussev, and C.Verbovski "Detecting stealth software with strider ghostbuster," In *Proceedings of the 2005 International Conference On Dependable Systems and Networks*, pp. 368–377, 2005.
- [17] H. Darabian, S. Homayounoot, A. Dehghantanha, S. Hashemi, H. Karimipour, R. M. Parizi, and K. K. R. Choo, "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, vol. 18, pp. 293–303, 2020.
- [18] T.M.Mitchell et al., "Machine learning," McGraw-hill New York, 2007, vol.1.
- [19] E.Beauxis-Aussalet and L.Hardman, "IEEE Conference on Visual Analytics Science and Technology (VAST)-Poster Proceedings," 2014, pp. 1–2.
- [20] N.Farnaaz and M.Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, Elsevier, Jan.2016, vol.89, pp. 213–217.
- [21] G. S. Fuhnwi, M. Revelle and C. Izurieta, "Improving Network Intrusion Detection Performance : An Empirical Evaluation Using Extreme Gradient Boosting (XGBoost) with Recursive Feature Elimination," *2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC)*, Houston, TX, USA, 2024, pp. 1-8, doi: 10.1109/ICAIC60265.2024.10433805.
- [22] Demšar, Janez, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, Dec. 2006, vol.7, pp. 1-30
- [23] Xu, Z., Ray, S., Subramanyan, P and Malik, S., " Malware detection using machine learning based analysis of virtual memory access patterns," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2017, pp. 169-174.
- [24] Bozkir, A.S., Tahillioglu, E., Aydos, M. and Kara, I., "Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision," *Computers & Security*, Elsevier, 2021, vol.103, pp.102166.
- [25] Nissim, N., Lahav, O., Cohen, A., Elovici, Y. and Rokach, L., " Volatile memory analysis using the MinHash method for efficient and secured detection of malware in private cloud," *Computers & Security*, Elsevier, 2019, vol.87, p.101590.
- [26] Javaheri, D. and Hosseinzadeh, M., " A framework for recognition and confronting of obfuscated malwares based on memory dumping and filter drivers," *Wireless Personal Communications*, Springer, 2018, vol.98, pp.119–137.
- [27] McLachlan, G.J., " Mahalanobis distance," *Resonance*, vol.4, no.6, pp.20–26, 1999
- [28] Ahmed S, Lee Y, Hyun SH, and Koo I, " Mitigating the impacts of covert cyber attacks in smart grids via reconstruction of measurement data utilizing deep denoising autoencoders," *Energies*, vol.12, no.16, pp.3091, 2019, MDPI.