Malware Detection using Anomaly Detection Algorithms

Attaullah Buriro*, Arslan Rafi†, Muhammad Azfar Yaqub‡, Flaminia Luccio*
*Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University, Venezia, Italy
†Centre for Cybersecurity, FBK, Trento
‡Faculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy
Email: {attaullah.buriro, luccio}@unive.it*, arafi@fbk.eu†, muhammadazfar.yaqub@unibz.it‡

***Abstract*— Malware, a diverse category of software specifically engineered to compromise devices, poses a serious threat to the security of computer systems and networks. Traditional malware detection methods, such as signature-based or behavior-based, rely on predefined patterns or manual analysis of malware characteristics or behaviors. However, these methods are ineffective against new or unknown malware, as they cannot recognize malware that does not match the existing patterns or profiles. Machine learning (ML) methods, on the other hand, can learn from data to detect malware based on complex patterns, without requiring prior knowledge or human intervention. In this paper, we propose and apply an anomaly detection approach on Programmable Executable files to detect and prevent malware installation. We evaluated our approach on a publicly available dataset, namely, Blue Hexagon Open Dataset for Malware AnalysiS (BODMAS) dataset using three classifiers, KNearest Neighbor, Support Vector Machine, and Random Forest to identify anomalies in the PE files. RF outperformed its counterparts and yielded highest accuracy of 99.73% with *zero* False Positive Rate.**

***Index Terms*—Malware, Machine Learning, Anomaly Detection**

I. Introduction

Malware is a term that encompasses various types of software designed to harm devices or steal information for malicious purposes [1]. Malware can take various forms, such as viruses, worms, Trojans, ransomware, spyware, backdoor, and rootkits, each with different characteristics and behaviors [2]. It poses a serious threat to the security of computer systems and networks[1], as it can cause data loss, data theft, identity theft, or device damage, affecting both individuals and organizations [4].

Traditional malware detection methods, such as signature-based [5], or behavior-based [6] [7], rely on predefined patterns or manual analysis of malware characteristics or behaviors. However, these methods have shown to be ineffective against new or unknown malware, as they cannot recognize malware that does not match the existing patterns or profiles. Moreover, these methods are prone to false positives and false negatives, and require constant updates and maintenance [6] [7].

ML methods, on the other hand, can learn from data to detect malware based on complex patterns, without requiring prior knowledge or human intervention [8] [9]. Machine learning methods can be classified into supervised, unsupervised, and semi-supervised, depending on the availability and quality of the labeled data [10] [11]. Machine learning methods can also adapt to the changing nature of malware and improve their performance over time, by learning from new data and feedback [12].

Anomaly detection is a type of unsupervised machine learning method that aims to identify instances that deviate from the normal behavior or distribution of the data [13]. Anomaly detection can be useful for malware detection, as it can detect new or unknown malware that do not conform to the normal behavior or distribution of the benign software [14]. Anomaly detection can be applied to various types of data, such as network traffic [15], system calls [16], or executable files [17] [18]. Anomaly detection methods can use various techniques, such as statistical, distance-based, density-based, clustering-based, or classification-based [19] [20]. Statistical methods assume that the data follows a certain distribution, and use statistical tests to identify outliers. Distance-based methods measure the distance between instances, and use a threshold to determine if an instance is anomalous. Density-based methods estimate the density of the data, and use a threshold to determine if an instance is in a low-density region. Clustering-based methods group similar instances together, and use a threshold to determine if an instance belongs to any cluster. Classification-based methods use supervised or semi-supervised learning to train a model that can distinguish between normal and anomalous instances [21].

In this paper, we propose and apply an anomaly detection approach to detect and prevent malware installation on Programmable Executable (PE) files. PE files are the standard format for executable files on Windows operating systems, and they contain various information and metadata about the executable, such as headers, sections, imports, exports, resources, and certificates [22]. We use three classifiers, i.e., KNearest Neighbor (KNN), Support Vector Machine (SVM), and Random Forest (RF), to identify anomalies in the PE files, based on the features extracted from the PE headers and sections. We evaluate our approach on the BODMAS dataset, which consists of more than 134k samples of benign and malicious PE files. Our experimental results show that our approach achieves a high accuracy of 99.73% using RF.

The main contributions of this paper are listed below:

[1] A network, in the context of information technology, is defined as the connection of at least two computer systems, either by a cable or a wireless connection. They are fundamental to the functioning of the modern digital world, supporting everything from small home Wi-Fi setups to global internet connectivity [3]

- The proposal and evaluation of an anomaly detection approach to detect and prevent malware installation on PE files, which are the standard format for executable files on Windows operating systems.
- Experimental evaluation of three anomaly detectors to identify anomalies in the PE files, based on the features extracted from the PE headers and sections, which are rich sources of information and metadata about the executable.
- The evaluation of scheme on a public dataset - BODMAS, which consists of more than 134k samples of benign and malicious PE files, and the demonstration that our approach achieves a high accuracy of 99.73% using RF.

The rest of the paper is organized as follows: Section II reviews the related work on malware detection and anomaly detection. Section III briefly explains our adapted approach of malware detection. Section IV describes the methodology of our anomaly detection approach, including the data preprocessing, the feature extraction, the anomaly detection algorithms, and the evaluation metrics. Section V presents and analyzes the experimental results of our approach, and compares them with the baseline methods. Section VI concludes the paper and suggests some directions for future research.

II. Related Work

ML-based methods have widely been applied to malware detection domain [10] [11], as they can learn patterns from data to detect malware without requiring prior knowledge or human intervention. Supervised learning uses labeled data for training models to classify instances as benign or malware, achieving high accuracy but requiring large labeled datasets, while unsupervised learning groups similar instances without labels and can detect new malware but with comparatively lower accuracy and interpretability. Semi-supervised learning leverages a small labeled and large unlabeled dataset to train classifiers with high accuracy using less labels. Technically speaking, anomaly detection aims at identifying outliers deviating from normal data distribution and can be useful for malware detection by finding new malware not conforming to benign distributions, by applying to network traffic, system calls, or files using statistical, distance-based, density-based, clustering-based, or classification-based techniques [13] [14] [15] [16] [17] [18] [19] [20] [21]. Furthermore, the anomaly detection approaches are one class classification algorithms, which are trained on only benign data samples and are tested against all kinds of traffic, i.e., benign and malware data samples.

The authors of [13] explored the potential application of ML algorithms for anomaly-based malware detection. They argue that unsupervised techniques like k-nearest neighbors (kNN) with $k = 4$ could effectively cluster normal program behaviors, identifying outliers indicative of malware. By utilizing static and dynamic analysis tools like PEview[2] in sandbox environment, the study argues that a wide range of discriminative features could be extracted to reflect malware functionality on the binary and behavioral levels.

The paper [23] addresses a crucial gap in the literature that is how to maintain the performance of different classifiers over time (i.e., concept drift). The authors use the EMBER dataset, which contains 1 million samples with temporal information, to train and test seven classifiers, including LightGBM, RF, KNN, and Decision Tree models. They evaluate the performance of these classifiers using metrics such as accuracy, AUC, precision, and recall, both at the initial training stage and over time. The paper reports that tree-based models, especially LightGBM and RF, exhibited the best performance stability over time. The authors also demonstrate that periodic re-training of the classifiers can help preserve their performance levels. Furthermore, the paper explores the use of AdaBoosted RF as a meta-model, which showed comparable or superior performance to LightGBM when applied to data from subsequent months. The paper concludes that machine learning models need to be re-trained periodically to cope with the changing nature of malware threats.

We consider the following papers extremely relevant to our study: Firstly, the study [22] evaluates different machine learning techniques for detecting malware in PE files. It uses the EMBER2018 dataset to train and evaluate the chosen models, including Light Gradient Boosting Machine (LGBM), RF, Logistic Regression (LR), Neural Networks. The experimentation with various feature sets extracted from the PE files, shows that Neural Networks consistently outperform LGBM baselines when all nine features are used, achieving an accuracy of 95.22%. The training is performed on a labeled dataset of 300,000 files from the EMBER dataset, while the evaluation is done on an separate test set of 100,000 files. The paper formulates malware detection as a binary classification task (distinguishing among files and assigning label 0 to benign and label 1 to malicious ones), and employs multiple classifiers for binary analysis. Secondly, the study [24] introduces MALWD&C that can detect and categorize malware with higher accuracy. MALWD&C is validated empirically on a public dataset, called, BODMAS, to demonstrate its efficacy. The study analyzes the important features and their influence on the performance of the method. The paper formulates the problem of malware detection and categorization as two related machine learning tasks: binary classification for distinguishing between malicious and benign files, and multi-class classification for assigning malware to different categories. It employs common machine learning classifiers, such as RF, Gradient Boosting (GB), Naive Bayes (NB), KNN, and Neural Networks, and reports that RF achieved the best performance for both tasks, with 99.56% accuracy for malware detection, and 97.69% accuracy for malware categorization, using the set of selected features. The paper also shows that feature selection improved the performance and reduced the training/testing time of the method, without compromising the accuracy.

We also consider the following papers in our related

[2] http://wjradburn.com/software/

work as these schemes have proposed and used one class classification for malware detection similar to our approach. The paper [13] discusses the use of KNN as an unsupervised learning algorithm for anomaly detection of malware. Authors argue that KNN is well-suited for one-class classification of malware, since it can perform the task without examples of "normal" (benign) files, just by finding distances of new samples from existing ones. The authors of [25] propose the use of One-Class SVDD and SVM for malware detection. They evaluated their approach on synthetic datasets containing mixtures of Gaussians, circles, and banana-shaped data, as well as a real-world Microsoft Malware Classification dataset, using binary and assembly code features with assembly code as privileged info. Obtained results demonstrate that training the one-class classifiers with the additional privileged information significantly improves their ability to accurately define the normal class over not using privileged info.

In this paper, we present a one-class classification approach for malware detection, which differs from the existing methods that mostly rely on binary [26] [24] or multi-class classification [27]. Our approach only requires benign samples for training and can detect unknown or zero-day malware without any prior knowledge. We have evaluated our approach on a large and diverse dataset of 134k executable files from BODMAS [27], which is significantly larger than the datasets used by the state-of-the-art anomaly detection methods. Our experimental results show that our approach achieves an accuracy of 99.73% with *zero* false acceptance rate of malware, demonstrating its effectiveness and robustness in malware detection. Our work opens up new possibilities and challenges for applying one-class classification in malware analysis and other security domains.

III. One Class Classification based Malware Detection Scheme

A. Portable Executable Files (PE)

PE format is the standard file format for Windows executables, DLLs, and other dynamic link libraries [28]. The PE format has several features and implications for malware analysis: Firstly, the PE format encapsulates the executable code and provides the instructions for loading the file into memory. These instructions include the entry point, code sections, resources, and other components of the program. Secondly, the PE format consists of two main parts: the PE header and the PE sections. The PE header contains information such as the location of libraries, the size of the stack, and the characteristics of the file. The PE sections contain the actual data and code of the program, such as initialized data (.data), read-only data (.rdata), and uninitialized data (.bss). These are some of the common section names, but they can vary depending on the compiler and linker used. Thirdly, the PE header also includes metadata in the optional headers, such as the machine type, the number of sections, and the timestamps. The section table lists the sections and their attributes, such as the name, the virtual address, the raw size, and the permissions. The structure of PE files is illustrated in Figure 1. Finally the PE format is portable across different

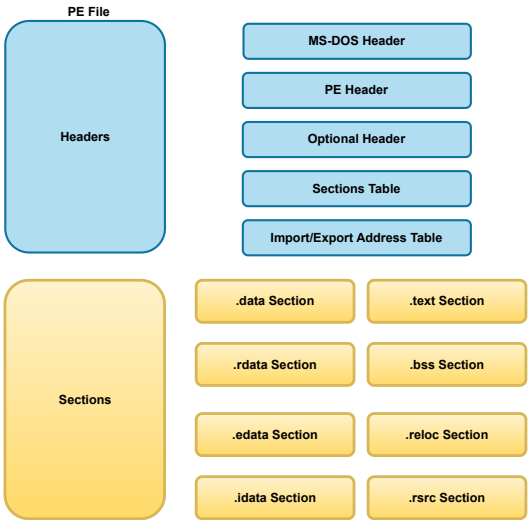

Fig. 1: PE file structure [28]

Windows systems, as it preserves the same basic structure regardless of the platform. This allows code and libraries to be reused and interchanged on various Windows versions. However, this also makes the PE format a very common malware vector, as malware can disguise itself as a normal Windows executable and evade detection. In fact, VirusTotal reported that nearly half of all files scanned in 2016 were PE files [29].

B. Why One Class Classification?

One-class classification (OCC) is a ML technique that aims to identify the instances of a specific class (e.g., benign here) among all possible instances, by learning from a training set that contains only the benign samples. OCC is also known as anomaly or novelty detection, as it can detect the instances that deviate from the benign or expected behavior of the target class.

OCC is more realistic in malware detection scenarios because it can overcome the challenges of obtaining and labeling a representative sample of all types of malware, which are constantly evolving, with up to 450k new instances designed daily[3], and becoming more sophisticated. Moreover, traditional binary or multi-class classification methods may not be able to distinguish between new or unknown malware and benign programs, resulting in high false negative rates [30]. Therefore, OCC can be a better alternative for malware detection, as it can learn the normal behavior of benign programs from a large and diverse dataset, and then flag any deviation from this behavior as potential malware [31].

C. Our Approach

Our approach to malware detection is based on one-class classification, which is a ML technique that learns from a single class of data and detects any deviation from it. Figure

[3] https://www.splunk.com/en_us/blog/learn/malware-detection.html

2 illustrates the training and testing phases of our approach. In the training phase, we only use the features of benign samples (green) to train the classifier. In the testing phase, we apply the classifier to new samples, which could be either benign (green) or malicious (red). The classifier's task is to learn the distribution of the benign data points and draw a boundary around them. Then, it compares the distribution of the test sample with the distribution of the training data and decides whether the test sample is benign or malicious. If the test sample is benign, the installation is allowed; otherwise, it is rejected.

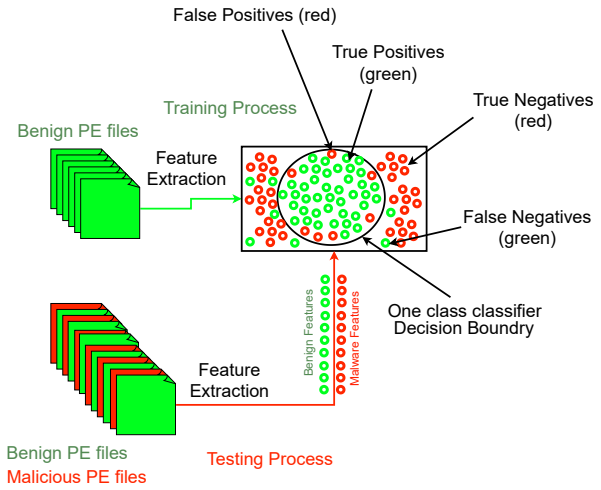

Fig. 2: Model diagram

IV. Experimental Evaluation

In this section we explain our methodology of conducting experiments.

A. Dataset

We used the BODMAS [27] dataset to evalaute our approach. The BODMAS dataset is a collection of 57,293 malware and 77,142 benign PE samples that were gathered between 2019 and 2020. The BODMAS dataset contains the samples from 581 malware families.

B. Features Extraction & Selection

We used Library to Instrument Executable Formats (LIEF)[4] to get features from executable files for our approach. LIEF is a Python library that can work with different types of executable files, such as PE. Using LIEF, we managed to extract 2381 features (similar to original features appeared in BODMAS [27] and EMBER [26] datasets) for each of the observations. We constructed features vectors, i.e., 2381 features long, to train our chosen ML models.

Feature subset selection is the technique of choosing the most relevant features for training ML models. Feature selection has two main benefits: firstly, it reduces the training time of the selected ML classifiers compared to the full set of features. Secondly, it lowers the computational cost without compromising accuracy. We used the same subset of features as identified in [24] for our experiments.

[4]https://github.com/lief-project/LIEF/releases

C. Classifiers Selection

Classifiers are ML models/algorithms that can learn from data and assign labels to new samples. There are many types of classifiers, such as decision trees, support vector machines, neural networks, etc. Each classifier has its own strengths and weaknesses, and may perform differently on different datasets or tasks. Classifier selection is an important step in machine learning, as it can affect the accuracy, efficiency, and robustness of the final model. There is no universal rule for classifier selection, as it depends on the characteristics of the data, the task, and the available resources. Therefore, classifier selection often requires experimentation and evaluation to find the optimal solution.

We chose three state-of-the-art machine learning anomaly detectors, i.e., LIBSVM (with one-class classification kernel), KNN, and RF for our analysis.

D. Classification Protocol

We conducted our experiments using one-class classification settings, as shown in Figure 2. We used 38571 labeled benign samples (positive class only) to train our selected classifiers and evaluated them on a test set of 38571 benign and 57293 malicious samples. Thus, the training set size was 38571 and the test set size was 95864. We applied 5-folds cross-validation on the training set to estimate the training error. The training error was crucial to compare with the test error in order to assess the overfitting of the classifiers. The smaller the gap between these two errors, the less overfitting the classifiers exhibited.

E. Parameter Optimization

Classifier's parameter optimization is the process of finding the optimal values of the parameters that affect the performance of the classifiers. There are different methods and techniques for parameter optimization of classifiers, such as grid search, random search, cross-validation, Bayesian optimization, evolutionary algorithms, meta-learning, and more. These methods vary in their complexity, speed, accuracy, and scalability, and they have different advantages and disadvantages [32]. The choice of the best method depends on the characteristics of the dataset, the classifier, the parameter space, and the evaluation criteria [32] [33].

LibSVM [34] a popular library for SVMs. Preliminary analyses proved polynomial kernel as the most suitable kernel for our dataset, as such, we tune the two parameters, ν and $degree$. The ν parameter is the most important one to tune for one class classification with LibSVM, as it affects the sensitivity and specificity of the model [34] [35]. A lower ν value means a larger decision boundary and a lower fraction of outliers, while a higher ν value means a smaller decision boundary and a higher fraction of outliers. The optimal ν value depends on the characteristics of the data and the desired level of robustness. The $degree$ parameter determines the $degree$ of the polynomial kernel function.

TABLE I: PARAMETER OPTIMIZATION OF SELECTED CLASSIFIERS

Classifiers	Parameters	Best Parameters	Highest Validation Accuracy(%)
LibSVM	ν = 0.1 to 1 with a step of 0.1 degree = 1 to 10 with a step of 1	ν = 0.1 degree = 3	82.043
KNN	k = 1 to 50 with a step of 5	k = 5	89.894
RF	no_of_trees = 100 to 600 with a step of 50	100	99.790

A higher $degree$ value means a more complex and flexible kernel function, while a lower $degree$ value means a simpler and smoother kernel function. The optimal $degree$ value depends on the shape and complexity of the data distribution. Similarly, for KNN and RF, we tried to optimize k which is number of nearest neighbors and no of trees, respectively.

We performed parameter optimization on the train set and used 5-fold cross validation to obtain the accuracy across different combination of hyperparameters (see Table I).

F. Performance Metrics

We use the following performance metrics to evaluate the performance of our approach and the baseline methods:

- **True Positive Rate (TPR)**: This metric measures the percentage of benign files that are correctly classified as benign by the classifier.
- **False Negative Rate (FNR)**: This metric measures the percentage of benign files that are incorrectly classified as malicious by the classifier.
- **False Positive Rate (FPR)**: This metric measures the percentage of malware files that are incorrectly classified as benign by the classifier.
- **True Negative Rate (TNR)**: This metric measures the percentage of malware files that are correctly classified as malware by the classifier.
- **Accuracy**: This metric measures the percentage of correctly classified files among all the files in the test set.

V. Results & Discussion

We use three state-of-the-art machine learning classifiers, namely, LibSVM, KNN, and RF as one-class classifier for our approach. We chose these classifiers based on our visual inspection of our dataset and their performance in previous studies for malware detection [13] [15] [24] [25] [26]. We exploited the same original features as [24] [26], however, we relied on the set of features found productive by Buriro et al. [24]. Hence the final feature vector is 25 features long.

All the classifiers were trained only on 38571 benign samples and were tested on combination of 38571 benign files (to obtain TPR and FNR) and 57,293 malware files (to obtain FPR and TNR). RF classifier outperformed its counterparts both in training and testing. It achieved validation and testing accuracy of 99.79% (compared to 88.73% and 82.043%, respectively, by KNN and LibSVM), and 99.73% (compared to 89.894% and 84.869%, respectively, by KNN and LibSVM), respectively. It is noteworthy that the all the malware samples were correctly rejected by the classifier hence we report *zero%* FPR. We obtained some false negative (102 samples) as though, which could cause unnecessary alerts and inconvenience for the users, however, the cost of wrong decision is less compared to getting a malware accepted as a benign file. It is also notable that the computed validation and testing accuracies are extremely close. This fact testifies that the training and testing errors are close too, negating overfitting.

RF classifier, as demonstrated, is a suitable choice for one-class classification, as it can handle both low and high-dimensional and heterogeneous data, capture complex and non-linear patterns, avoid overfitting and improve generalization, and handle imbalanced and noisy data. These properties enable the RF classifier to learn the characteristics of the benign class well and detect any deviation from the normal behavior as an anomaly. Moreover, the RF classifier is scalable and adaptable, as it can handle different types of features and data distributions. Therefore, the RF classifier works well for one-class classification, especially for malware detection, where the data is often large, diverse, and dynamic.

VI. Conclusion & Future Work

In this paper, we have presented a one-class classification approach for malware detection, which differs from the existing methods that mostly rely on binary or multi-class classification. Our approach only requires benign samples for training and can detect unknown or zero-day malware without any prior knowledge. We have evaluated our approach on a large and diverse dataset of 134k executable files from BODMAS, which is significantly larger than the datasets used by the state-of-the-art anomaly detection methods. Our experimental results show that our approach achieves an accuracy of 99.73% using RF as one-class classifier with *zero* FPR of malware, demonstrating its effectiveness and robustness in malware detection.

As future work, firstly, we plan to extend our approach to other types of malware, such as Android malware, web malware, etc. This would require collecting and analyzing more data from different sources and platforms, and adapting our feature extraction and classification methods accordingly. Secondly, we plan to incorporate other types of features, such as image features, etc. This would require exploring and developing new techniques for extracting and representing such features, and evaluating their impact on the performance of our approach. Thirdly, to explore other types of one-class classifiers, such as Deep Neural Networks (DNN), generative models, etc. This would require investigating the advantages and disadvantages of such classifiers, and comparing them with the best performing classifier, i.e., RF. Fourthly, we plan to investigate the trade-off between the detection accuracy and the computational efficiency of our approach. This would require measuring and optimizing the time and space complexity of our approach, and finding the

optimal balance between the quality and the speed of the detection. Finally, we plan to evaluate our approach on real-world scenarios and datasets, such as IoT networks, cloud services, etc. This would require collaborating with industry partners and practitioners, and testing our approach on large-scale and dynamic data streams.

ACKNOWLEDGMENT

Authors would like to thank all the volunteers for their participation in the experiments and groupmates for their valuable and insightful thoughts.

This work is partially supported by projects "SEcurity and RIghts In the CyberSpace - SERICS" (PE00000014 - CUP H73C2200089001), "Interconnected Nord-Est Innovation Ecoscheme - iNEST" (ECS00000043 - CUP H43C22000540006), and PRIN/PNRR "Automatic Modelling and ∀erification of Dedicated sEcUrity deviceS - AM∀DEUS" (P2022EPPHM - CUP H53D23008130001), all under the National Recovery and Resilience Plan (NRRP) funded by the European Union - NextGenerationEU.

REFERENCES

[1] What are the different types of malware? [Accessed: Dec. 20, 2023]. [Online]. Available: https://www.kaspersky.com/resource-center/threats/types-of-malware

[2] Types of malware and how each impacts your computer. [Accessed: Dec. 20, 2023]. [Online]. Available: https://www.technology-solved.com/types-of-malware/

[3] B. Liu, Z. Yan, and C. W. Chen, "Medium access control for wireless body area networks with qos provisioning and energy efficient design," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 422–434, 2017.

[4] Malware. [Accessed: Dec. 20, 2023]. [Online]. Available: https://www.cyber.gov.au/threats/types-threats/malware

[5] Malware detection top techniques today. [Accessed: Dec. 22, 2023]. [Online]. Available: https://www.splunk.com/en_us/blog/learn/malware-detection.html

[6] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 12, pp. 59–67, 2016.

[7] H.-Y. Kwon, T. Kim, and M.-K. Lee, "Advanced intrusion detection combining signature-based and behavior-based detection methods," *Electronics*, vol. 11, no. 6, p. 867, 2022.

[8] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware detection using machine learning and deep learning," in *Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings 6*. Springer, 2018, pp. 402–411.

[9] V. Dhingra, J. Singh, and P. Kaur, "Detecting and analyzing malware using machine learning classifiers," in *International Conference on Next Generation Systems and Networks*. Springer, 2022, pp. 197–207.

[10] Y. Chen, M. Mancini, X. Zhu, and Z. Akata, "Semi-supervised and unsupervised deep visual learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[11] Introduction to supervised, semi-supervised, unsupervised and reinforcement learning. [Accessed: Dec. 24, 2023]. [Online]. Available: https://www.baeldung.com/cs/machine-learning-intro

[12] M. Aslam, D. Ye, M. Hanif, and M. Asad, "Adaptive machine learning: A framework for active malware detection," in *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2020, pp. 57–64.

[13] S. Rani, K. Tripathi, Y. Arora, and A. Kumar, "Analysis of anomaly detection of malware using KNN," in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, vol. 2. IEEE, 2022, pp. 774–779.

[14] D. Escudero García and N. DeCastro-García, "Application of anomaly detection models to malware detection in the presence of concept drift," in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 2023, pp. 15–26.

[15] T. Nishio, M. Nakahara, N. Okui, A. Kubota, Y. Kobayashi, K. Sugiyama, and R. Shinkuma, "Anomaly traffic detection with federated learning toward network-based malware detection in IoT," in *IEEE Global Communications Conference (GLOBECOM 2022)*. IEEE, 2022, pp. 299–304.

[16] S. Shakya and M. Dave, "Analysis, detection, and classification of android malware using system calls," *arXiv preprint arXiv:2208.06130*, 2022.

[17] T.-L. Wan, T. Ban, S.-M. Cheng, Y.-T. Lee, B. Sun, R. Isawa, T. Takahashi, and D. Inoue, "Efficient detection and classification of internet-of-things malware based on byte sequences from executable files," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 262–275, 2020.

[18] N. Jadvani, M. Agarwal, and K. Leelasankar, "Malware detection based on portable executable file features," in *International Conference on Computing, Communication, Electrical and Biomedical Systems*. Springer, 2022, pp. 377–384.

[19] M. Fahim and A. Sillitti, "Anomaly detection, analysis and prediction techniques in IoT environment: A systematic literature review," *IEEE Access*, vol. 7, pp. 81 664–81 681, 2019.

[20] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine learning for anomaly detection: A systematic review," *IEEE Access*, vol. 9, pp. 78 658–78 700, 2021.

[21] D. Samariya and A. Thakkar, "A comprehensive survey of anomaly detection algorithms," *Annals of Data Science*, vol. 10, no. 3, pp. 829–850, 2023.

[22] C. Connors and D. Sarkar, "Machine learning for detecting malware in PE files," *arXiv preprint arXiv:2212.13988*, 2022.

[23] C. Galen and R. Steele, "Evaluating performance maintenance and deterioration over time of machine learning-based malware detection models on the ember PE dataset," in *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2020, pp. 1–7.

[24] A. Buriro, A. B. Buriro, T. Ahmad, S. Buriro, and S. Ullah, "Malwd&c: A quick and accurate machine learning-based approach for malware detection and categorization," *Applied Sciences*, vol. 13, no. 4, p. 2508, 2023.

[25] E. Burnaev and D. Smolyakov, "One-class svm with privileged information and its application to malware detection," in *2016 IEEE 16th International conference on data mining workshops (ICDMW)*. IEEE, 2016, pp. 273–280.

[26] H. S. Anderson and P. Roth, "Ember: an open dataset for training static PE malware machine learning models," *arXiv preprint arXiv:1804.04637*, 2018.

[27] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "Bodmas: An open dataset for learning based temporal analysis of PE malware," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 78–84.

[28] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020.

[29] A. Kumar, K. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *Journal of King Saud University-Computer and Information Sciences*, vol. 31, no. 2, pp. 252–265, 2019.

[30] M. Al-Qudah, Z. Ashi, M. Alnabhan, and Q. Abu Al-Haija, "Effective one-class classifier model for memory dump malware detection," *Journal of Sensor and Actuator Networks*, vol. 12, no. 1, p. 5, 2023.

[31] A. Tajoddin and M. Abadi, "Ramd: registry-based anomaly malware detection using one-class ensemble classifiers," *Applied Intelligence*, vol. 49, pp. 2641–2658, 2019.

[32] M. Reif, F. Shafait, and A. Dengel, "Meta-learning for evolutionary parameter optimization of classifiers," *Machine learning*, vol. 87, pp. 357–380, 2012.

[33] Q. Wang and H.-L. Chan, "Generating pool of classifiers with hyper-parameter optimization for ensemble," in *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2021, pp. 1–6.

[34] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[35] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine learning*, vol. 46, pp. 131–159, 2002.