IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁑ Impact Factor 8.102 ⁑ Vol. 12, Issue 4, April 2023

DOI: 10.17148/IJARCCE.2023.12435

Machine Learning in Malware Detection: A Survey of Analysis Techniques

Raghavendra R[1], Vikram Dutta M[2]

Professor, MCA Department (School of CS & IT), Jain University, Bangalore, India[1]

Student, MCA Department (School of CS & IT), Jain University, Bangalore, India[2]

Abstract: In the contemporary digital era, the malware presents a substantial challenge to internet users, particularly with the rise of polymorphic malware that persistently alters its discernible features to evade detection by traditional signature-based models. This advanced malware variety, which boasts a dynamic design and inherits traits from multiple malware types, differs significantly from its conventional counterparts. The proposed research aims to examine and analyse the behaviour of malware executables, mainly focusing on their polymorphic attributes, to enhance cybersecurity through better understanding and detection.

Tackling the growing complexity and volume of malicious software is arduous, prompting researchers to employ machine learning techniques to decipher the underlying patterns and models within this intricate landscape, thereby keeping pace with malware's continuous evolution. This comprehensive review sheds light on using machine learning in the context of malware analysis for Windows environments, explicitly targeting Portable Executables.

In our analysis of the existing literature, we classify the studies based on their objectives, the malware-specific data and features, and the machine-learning approaches they apply. Furthermore, we delve into the challenges and issues tied to dataset usage and identify the dominant trends and promising future directions.

Keywords: Machine Learning, Cyber Security, Malware, Malware Analysis

I. INTRODUCTION

Examining malicious executables is essential in computer security as the prevalence and sophistication of malware attacks continue to grow. Technological advancements necessitate the adoption of cutting-edge machine-learning algorithms for the practical identification and classification of malicious executables. This subject explores the application of various machine-learning algorithms to classify harmful executables. We will delve into the benefits and drawbacks of each algorithm and assess its efficacy in detecting malicious files. This analysis aims to provide an analysis of multiple machine learning algorithm which are used for classifying harmful executables and offer valuable insights to inform the creation of more efficient malware detection systems.

II. REVIEW ON RELEVANT IMPLEMENTATIONS

In paper [1], the authors introduce a defensive mechanism employing machine learning (ML) algorithms for malware identification. They assess three ML approaches to malware detection, selecting the most suitable one. The decision tree (DT) algorithm achieves the highest accuracy at 99%, followed by convolutional neural networks (CNN) at 98.76% and support vector machine (SVM) at 96.41%. The study compares the three algorithms' false favorable rates (FPR), with DT exhibiting the lowest FPR at 2.01%. The research uses a dataset from the Canadian Institute for Cybersecurity to train, test, and compare the effectiveness of the three ML models (DT, CNN, and SVM). The study concludes that ML algorithms can accurately differentiate malicious and benign data using static analysis based on PE information and carefully chosen data. The study highlights the advantage of not needing to execute anything to determine if data are malicious. The results show that the DT ML method achieved the highest accuracy of the classifiers evaluated.

In paper [2], the researchers aim to assess the effectiveness of various ML algorithms for classifying malicious executables by applying them to static features extracted from legitimate and malicious executable files. The results indicate that the Decision Tree algorithm has an accuracy of 99.14%, Random-forest has an accuracy of 99.47%, and Light Gradient Boosting Machine (Light GBM) has the highest accuracy of 99.50%. The study finds that Light GBM is the most efficient in terms of accuracy and time taken to train the model and has the lowest False Negative rate. However, the study also acknowledges that there are areas for further research, and new techniques are needed to tackle new malware as internet use is growing.

DOI: 10.17148/IJARCCE.2023.12435

In paper [3], the researchers propose a novel approach called fine-grained dangerous permission (FDP) for identifying malicious Android applications. The method employs features that effectively distinguish between malicious and benign applications. The experimental results show that the FDP method, implemented using K-Nearest Neighbour (KNN) classifier, has a higher detection rate of Android malicious families than existing methods. Moreover, the FDP method is efficient enough for practical implementation in Android malware detection. However, the study also acknowledges that the FDP approach has limitations, as it cannot extract information from techniques such as dynamic loading, reflection mechanisms, and encryption due to the inherent limitations of static analysis. The study suggests incorporating dynamic analysis methods to extract additional relevant features, significantly increasing the detection overhead.

In paper [4], the researchers present a new lightweight malware detection system called TinyDroid, developed by the paper's authors. The system uses reverse engineering to extract Dalvik instructions from DEX files and simplifies them into a small symbol set. Then, a detection model is established using N-gram, exemplar selection method, and ML algorithms such as KNN, SVM, Naive Bayes, and Random Forest. The results show that the Random Forest Classifier outperforms other classifiers by achieving the highest accuracy scores for all n-Gram sequences. The system is evaluated against real-world antivirus scanners, and it is found that TinyDroid performs better than the state-of-the-art tools in detection and classification. However, the paper also points out some limitations and deficiencies in the study, such as the samples mainly coming from academic sample sets and the need for metamorphic malware samples.

The paper [5] introduces a malware detection system for Android devices that employs a support vector machine (SVM) as the ML algorithm. The system is designed as an alternative to traditional detection methods and aims to detect unknown Android applications. The proposed system extracts feature from the applications using static and dynamic analysis using the Principle Component Analysis Method (PCA).

In the paper [6], the researchers introduce two new static API call datasets, VirusShare and VirusSample, consisting of MD5 hash-codes, static API calls, and malware families. The authors apply simple preprocessing by removing malware families with fewer than 100 occurrences and samples marked as undefined by VirusTotal. Additionally, to address the imbalanced distribution of malware families, balanced versions of the two datasets were created by imposing an upper limit of 300 for the malware families. The benchmark results of various models, including two traditional machine learning algorithms, two ensemble models, a recurrent neural network, and two pre-trained transformer models, were presented for the imbalanced and balanced dataset versions.

For each dataset, the authors employed a Stratified 5-Fold strategy on the training data, with 20% of the training data used for validation in each iteration. The results reveal that SVM outperforms other models in F1-score for both versions of the VirusShare dataset, while XGBoost achieves the highest AUC score for each dataset. Conversely, in the VirusSample dataset, LSTM and CANINE obtain the highest F1-score in imbalanced and balanced versions, respectively. The authors anticipate that these two datasets will allow researchers to test their methods and compare the presented results.

In the paper [7], the authors conduct comparative experiments to assess the effectiveness of their feature extraction method, which is based on local maliciousness extraction API fragments for detecting malicious code. They discover that their method and the second and third methods are more accurate than the method based on the overall API execution sequence. The authors also find that their LSTM-based model outperforms the CNN and SVM models.

The authors' method exhibits anti-interference ability, an essential feature for practical application in malware detection. Their method is innovative as it analyses the local maliciousness of malicious code, and their approach effectively investigates the application of deep learning in this area. The authors plan to explore the combination of prior knowledge and deep learning in future work. Overall, the authors' method is effective and practical in detecting malware.

III. MALWARE ANALYSIS AND DETECTION: BACKGROUND

Static Analysis

This approach involves examining software without executing it. Reverse engineering is a technique used in static evaluation to deconstruct the process [8]. By extracting the strings, Entropy, and image headers, static evaluation compares the program to a vast database of signatures [9]. While this method can quickly analyze and identify known malware, it struggles with complex and new malware variants. Advanced static evaluation methods can analyze intricate malware, but these processes are often laborious and demand extensive knowledge of operating systems and disassembly.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁙ Impact Factor 8.102 ⁙ Vol. 12, Issue 4, April 2023

DOI: 10.17148/IJARCCE.2023.12435

Dynamic Analysis

This type of analysis focuses on the attributes of executed strategies while the program is running. Various methods can be employed in dynamic evaluation, including monitoring function calls, analyzing function parameters, tracking information flow, tracing instructions, and examining AutoStart Extensibility Points [2]. Changes in files, registry, network activities, and access to RAM and hard drives can also be scrutinized for dynamic malware analysis. For detecting new malware, dynamic evaluation is typically preferred over static evaluation, as the behaviour and characteristics of malware remain consistent even when their structure changes [9]. However, modern malware has become more sophisticated and may cease execution when it detects behavioural analysis. In such cases, machine learning outperforms dynamic and static evaluation methods [6].

Signature-Based Detection Method:

Signatures are distinct attributes of a malware executable, comparable to a fingerprint [8]. Antivirus applications have extensively used this approach. While faster than other methods, it struggles with identifying new malware [15, 16].

Behaviour-Based Detection Method:

This method evaluates a program's behaviour to ascertain if it is malicious or benign [10]. It inspects the program's activities rather than its code or sequences. Despite code alterations, the behaviour stays consistent or similar, enabling detection via this method [11]. A notable limitation of behaviour-based detection is that some malware may not exhibit all behaviours in a secure environment (e.g., virtual machine or sandbox), resulting in possible false negatives [12]. Typically, this method comprises three stages: extracting behaviours, establishing properties through related behaviour identification, and using machine learning or data mining techniques for sample classification as malicious or benign.

Heuristic-Based Detection Method:

Heuristic approaches depend on experience and machine learning techniques to pinpoint characteristic features of relevant programs. By analysing these features, heuristics can detect malware with comparable traits [13], [14]. The heuristic method is initially trained with several samples, after which the trained system examines new program samples to identify malware. Features employed in the heuristic method encompass application programming interface (API) system calls, operational code (Opcode), N-Grams, control flow graphs (CFG), and hybrid features [15]. Heuristic methods can detect malware that signature- or behaviour-based methods may miss and can even recognize unknown malware to some degree. However, they might still face difficulties in detecting novel or intricate malware.

Opcode Sequence Analysis:

In paper [4], Opcode sequences extracted from DEX files are used as features for the malware detection system TinyDroid. OpCode sequence analysis involves analysing the sequence of operation codes (opcodes) that form part of the executable's low-level machine instructions. By examining these sequences, researchers can identify patterns and features that are indicative of malicious behaviour, leading to improved malware detection.

API Call Analysis:

The studies in paper [6] and [7] focus on analysing the local maliciousness of malicious code through API call analysis. API calls are functions provided by an operating system or library that allow an application to interact with the system or other software components. By examining the API calls made by an executable, researchers can identify patterns and features that may be indicative of malicious behaviour.

System Call Analysis:

Heuristic-based detection methods often involve analysing system calls made by an executable. System calls are requests made by an application to the operating system for services such as file access, memory management, and process control. By examining the patterns of system calls, researchers can identify features that are indicative of malicious behaviour.

Control Flow Graph Analysis:

Another feature extraction technique used in heuristic-based malware detection is the analysis of control flow graphs (CFGs). CFGs represent the flow of control within a program, illustrating the various paths that can be taken during execution. By analysing the structure of a program's CFG, researchers can identify patterns and features that are indicative of malicious behaviour.

Hybrid Feature Extraction:

In some cases, researchers may choose to employ a combination of feature extraction techniques to improve the effectiveness of malware detection. This approach, known as hybrid feature extraction, may involve combining static and dynamic analysis methods or using multiple feature extraction techniques simultaneously.

IV. UNDERSTANDING POPULAR FEATURE EXTRACTION STRATEGIES

PCA Feature Selection Method

In this paper [16] the author works on minimizing the number of features during the learning and classification stages. By employing feature selection and extraction methods alongside a support vector machine (SVM) classifier, a general dimension reduction technique is developed, allowing the learning model to be updated in mere seconds.

The n-gram feature space data, defined as

$$fN = (d, \omega N),$$

This can be integrated into a variety of learning algorithms. Originally, SVMs were developed for binary classification. Hsu in [17] devised a multiclass classifier by combining numerous binary classifiers. The one-against-one method mandates the resolution of a binary classification problem using training data from the ith and jth classes.

The efficacy of SVMs hinges on kernel selection, kernel parameters, and the soft margin parameter C. Maximizing hyperplane margins involves using a Gaussian radial basis function, denoted as:

$$k(xi, xj) = \exp(-\gamma||xi - xj||2).$$

To attain optimal predictions, kernel parameter γ and cost parameter C must be estimated. The test environment incorporated the LIBSVM tool

PCA and KPCA were introduced as feature extraction algorithms. Dimension reduction aims to cut down the time cost associated with machine learning. A variety of selection methods were utilized to choose feature subsets for learning and classification testing.

The Monotonic Groupwise Term Frequency-Inverse Document Frequency (MG TF-IDF) feature selection approach was employed by the author in [16], as the smallest number of features while maintaining comparable micro-precision levels. This makes it superior to the standard TF-IDF method for selecting effective features for individual malware families. By reducing the feature dimension to between 100 and 1000 features, equivalent micro-precision is maintained, saving up to 99% of time cost in high-dimensional feature spaces.

Both PCA and MG PCA were employed to effectively extract major malware behaviors, forming a limited number of representative components. The top five principal component score weightings and the combination of latent functions extracted with the PCA method were listed. The first principal component revealed an increase in registry key and memory operations, while the second principal component demonstrated an increase in file operations, aligning with general virus attack characteristics.

The MG KPCA method displayed a substantial enhancement in prediction accuracy. When the number of transformed features was minimal, MG KPCA attained greater micro-precision than other feature reduction techniques. However, increasing the number of transformed features led to overfitting and a decrease in micro-precision levels.

Hence it was observed that PCA and KPCA methods successfully reduced the time cost of learning and classification processes, they increased the time cost of the feature extraction process. In terms of reducing the time cost of online training, PCA and MG PCA methods were found to be the most effective.

N-Gram Feature Selection:

In their study, the researchers in [15] examined a dataset of 400,000 files from Group B, identifying 4,289,759,510 distinct 4-grams and 35,953,973,975 distinct 6-grams. Storing 6-grams using 32 or 64-bit integers for count data necessitates 503 or 791 GB of RAM, respectively, posing a significant obstacle for implementing byte n-gram features.

DOI: 10.17148/IJARCCE.2023.12435

Nonetheless, individual n-gram frequencies adhere to a power-law distribution, with 87.72% of 6-grams appearing only once, 97.58% appearing ten or fewer times, and 99.61% appearing 100 or fewer times. By selecting a minimum occurrence threshold based on the dataset's coverage, the pool of candidate n-grams can be decreased. For example, selecting 6-grams present in at least 1% of the 400,000 files results in nearly 1.6 million 6-grams, a reduction of more than 99.99%.

However, learning from 1.6 million 6-grams still presents computational challenges and overfitting risks. The researchers adopted a two-phase feature selection approach, initially conducting a coarse feature selection down to 200,000 n-grams, followed by a final feature selection during model building.

To determine the 200,000 n-gram subset, the researchers compared several ranking schemes, including Information Gain, Malice Score, Benign Score, Absolute Malice Score, Root Malice Score, Gini coefficient, and KL-divergence. Each method was assessed to select the initial 200,000 n-gram subset. The feature ranking methods' efficacy was evaluated based on their capacity to identify features predominantly present in one of the classes (malware or benign).

Past studies relied solely on feature ranking schemes for feature selection. A limitation of this method is the need to estimate the number of top-ranked features to incorporate into the model, which can result in overfitting or underfitting. The researchers aimed to overcome this issue and enhance the model's performance by employing a two-stage feature selection strategy.[18]

V. DISCUSSION

Based on the implementation strategies discussed above, several benefits and disadvantages can be identified.

Benefits:

Improved detection capabilities:

By employing various machine learning algorithms, feature extraction techniques, and hybrid analysis methods, researchers have developed more accurate and efficient malware detection systems. For instance, the FDP method in paper [3] and the TinyDroid system in paper [4] demonstrate higher detection rates for Android malware families compared to existing methods.

Adaptability to new malware:

Methods such as the SVM-based system in paper [5] and the API call analysis approach in paper [6] and [7] can detect unknown or new Android applications, enhancing the adaptability of malware detection systems to novel threats.

Reduced time cost:

Techniques like PCA and KPCA in paper [16] help in reducing the time cost of learning and classification processes, making malware detection systems more efficient and practical for real-world applications.

Comprehensive feature selection:

Strategies like N-Gram feature selection, as discussed in [15], offer a more thorough analysis of malware features, leading to improved detection capabilities.

Disadvantages:

Limitations of static analysis:

The FDP method in paper [3] faces challenges in extracting information from techniques like dynamic loading, reflection mechanisms, and encryption due to the inherent limitations of static analysis.

Limited real-world testing:

Some studies, like the TinyDroid system in paper [4], primarily rely on academic sample sets and may not have been tested against a broader range of real-world malware samples, including metamorphic malware.

High computational demands:

Some feature extraction techniques, like N-Gram feature selection in [15], can require extensive computational resources, making it challenging to implement them in resource-constrained environments.

Overfitting risks:

The use of certain feature extraction techniques, such as PCA and KPCA in paper [16], may lead to overfitting when the number of transformed features is increased, which can negatively impact the model's performance.

VI. CONCLUSION

In conclusion, the reviewed research papers provide significant contributions to the field of Android malware detection, employing various methodologies and techniques to enhance the identification of malicious applications. The fine-grained dangerous permission (FDP) method proposed in paper [3] demonstrates a higher detection rate of Android malicious families than existing methods, although the limitations of static analysis hinder its performance. The novel lightweight malware detection system, TinyDroid, developed in paper [4], shows promising results in comparison to real-world antivirus scanners.

The malware detection system introduced in paper [5] leverages an SVM as the ML algorithm and utilizes both static and dynamic analysis methods to detect unknown Android applications. In papers [6] and [7], new static API call datasets are introduced, and innovative feature extraction methods based on local maliciousness extraction API fragments are explored. These studies demonstrate the effectiveness and practicality of these approaches in malware detection.

Various malware analysis techniques, such as static and dynamic analysis, signature-based, behavior-based, and heuristic-based detection methods, are employed to improve the efficiency and accuracy of malware identification. OpCode sequence analysis, API call analysis, system call analysis, control flow graph analysis, and hybrid feature extraction strategies are used in heuristic-based detection methods to enhance the detection of malware. Popular feature extraction strategies, such as PCA and KPCA methods, and N-Gram feature selection, play a vital role in reducing time and computational costs while maintaining accurate detection rates.

The research papers discussed in this review highlight the advancements in Android malware detection techniques and methodologies. Although these approaches have proven to be effective, there is still room for improvement, particularly in addressing the limitations and deficiencies identified in each study. Future research should focus on incorporating dynamic analysis methods to counter the limitations of static analysis, examining real-world and metamorphic malware samples, and exploring the combination of prior knowledge and deep learning to develop more robust and accurate malware detection systems.

As malware continues to evolve and become more sophisticated, it is crucial for researchers to stay ahead of emerging threats by developing and refining malware detection methods. By considering the achievements and challenges presented in the reviewed research papers, the cybersecurity community can work collectively to create novel, efficient, and effective solutions to protect Android devices and their users from malicious applications. Ultimately, the pursuit of improved malware detection techniques will contribute to a more secure digital environment for all.

REFERENCES

[1] M. S. Akhtar and T. Feng, "Malware Analysis and Detection Using Machine Learning Algorithms," Symmetry (Basel), vol. 14, no. 11, p. 2304, Nov. 2022, doi: 10.3390/sym14112304.
[2] S. Agarkar and S. Ghosh, "Malware detection & classification using machine learning," in Proceedings - 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security, iSSSC 2020, Institute of Electrical and Electronics Engineers Inc., Dec. 2020. doi: 10.1109/iSSSC50941.2020.9358835.
[3] X. Jiang, B. Mao, J. Guan, and X. Huang, "Android Malware Detection Using Fine-Grained Features," Sci Program, vol. 2020, 2020, doi: 10.1155/2020/5190138.

[4] T. Chen, Q. Mao, Y. Yang, M. Lv, and J. Zhu, "TinyDroid: A lightweight and efficient model for android malware detection and classification," Mobile Information Systems, vol. 2018, 2018, doi: 10.1155/2018/4157156.
[5] L. Wen and H. Yu, "An Android malware detection system based on machine learning," in AIP Conference Proceedings, American Institute of Physics Inc., Jul. 2017. doi: 10.1063/1.4992953.
[6] B. Gençaydın and B. Düzgün, "Benchmark Static API Call Datasets for Malware Family Classification." [Online]. Available: https://github.com/khas-ccip/api
[7] X. Ma, S. Guo, W. Bai, J. Chen, S. Xia, and Z. Pan, "An API Semantics-Aware Malware Detection Method Based on Deep Learning," Security and Communication Networks, vol. 2019, 2019, doi: 10.1155/2019/1315047.

[8] D. Uppal, V. Mehra, and V. Verma, “Basic survey on Malware Analysis, Tools and Techniques,” International Journal on Computational Science & Applications, vol. 4, no. 1, pp. 103–112, Feb. 2014, doi: 10.5121/ijcsa.2014.4110.

[9] O. Aslan and R. Samet, “Investigation of possibilities to detect malware using existing tools,” in Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA, IEEE Computer Society, Mar. 2018, pp. 1277–1284. doi: 10.1109/AICCSA.2017.24.

[10] B. Panduri, M. Vummenthala, S. Jonnalagadda, G. Ashwini, N. Nagamani, and A. Akhila, “Dynamics and an efficient malware detection system using opcode sequence graph generation and ml algorithm,” in E3S Web of Conferences, EDP Sciences, Aug. 2020. doi: 10.1051/e3sconf/202018401009.

[11] G. Wagener, R. State, and A. Dulaunoy, “Malware behaviour analysis,” Journal in Computer Virology, vol. 4, no. 4, pp. 279–287, Nov. 2008, doi: 10.1007/s11416-007-0074-9.

[12] S. L. S. Darshan, M. A. A. Kumara, and C. D. Jaidhar, “Windows malware detection based on cuckoo sandbox generated report using machine learning algorithm,” in 11th International Conference on Industrial and Information Systems, ICIIS 2016 - Conference Proceedings, Institute of Electrical and Electronics Engineers Inc., Jul. 2016, pp. 534–539. doi: 10.1109/ICIINFS.2016.8262998.

[13] M. Mays, N. Drabinsky, and S. Brandle, “Feature Selection for Malware Classification.”

[14] H. X. Tencent, C. Eckert, C.-T. Lin, N.-J. Wang, and H. Xiao, “Feature Selection and Extraction for Malware Classification,” 2015. [Online]. Available: https://www.researchgate.net/publication/283129448

[15] E. Raff et al., “An investigation of byte n-gram features for malware classification,” Journal of Computer Virology and Hacking Techniques, vol. 14, no. 1, pp. 1–20, Feb. 2018, doi: 10.1007/s11416-016-0283-1.

[16] H. X. Tencent, C. Eckert, C.-T. Lin, N.-J. Wang, and H. Xiao, “Feature Selection and Extraction for Malware Classification,” 2015. [Online]. Available: https://www.researchgate.net/publication/283129448

[17] C.-W. Hsu and C.-J. Lin, “A Comparison of Methods for Multi-class Support Vector Machines.”

[18] M. Ali, S. Shiaeles, G. Bendiab, and B. Ghita, “Malgra: Machine learning and N-GRAM malware feature extraction and detection system,” Electronics (Switzerland), vol. 9, no. 11, pp. 1–20, Nov. 2020, doi: 10.3390/electronics9111777.