A Survey of AI-based Real-Time Malware Detection Systems

Tanzeel Ahmed
{tanzeelahmed713@gmail.com}

Abstract — **The internet has transformed the world into a giant database and has now become an essential part of our lives. The economical and financial shift of the industry toward IT has caused exponential growth in the malwares. The traditional techniques used by the majority of the Anti-Viruses for malware detection can be bypassed using different evasion techniques and are now less effective against the growing numbers and variants of malware. With the increase in malware threats, the data and privacy of the people are in danger, malware detection now demands an intelligent solution that can detect malware effectively and efficiently. Many AI models have been proposed to detect malwares. The performance of the AI models largely depends upon the features and the dataset used for training. In this survey, we reviewed several different models. We discussed the input features, the dataset used, model and the results of each paper that was studied. This survey will list the shortcomings of the proposed models and provide suggestions for the future work.**

Introduction:

Malware developers are now motivated more than ever to steal information, create false identities or steal credentials for impersonation. Over the years, we have created a massive database and now keep our information or knowledge on computers which has brought ease to our lives. However, it has made data theft easier than ever. Data have been stolen from more than 11 million internet users. A recent survey revealed that cyber theft has increased by 600 percent since last year [1], which shows that we are all at risk of getting our personal information stolen if it is not protected properly. Malware is malicious software that performs malicious activities like invading privacy, stealing money, causing harm, etc. According to AV-test statistics, over 450,000 malware and potentially unwanted applications (PUA) are registered every day [2]. The important term to note here is “registered”, which means these numbers only represent the malwares which get detected, not giving us the actual number of the malwares who succeeds to perform malicious activity. In 2022, Sonic Wall’s patented Real-Time Deep Memory Inspection™ (RTDMI) discovered 270,228 new malware variants — a 45% year-to-date increase, and an average of 1,501 new variants per day [3].

With new malware variants emerging daily, the traditional techniques to detect malware are less effective now. The two widely used techniques for malware detection are static and dynamic analysis. Static analysis involves signature-based detection. The signature of the binary file is calculated and compared with a set of known malware signatures. The drawback of this approach is that it can only detect known malware and can be bypassed obfuscation and code manipulation technique. On the other hand, dynamic analysis involves behavior-based detection which is more effective than static analysis. The binary file is executed in the controlled environment and its behavior is observed. However, this approach is not scalable and is prone to time delay attack.

With the emergence of Artificial Intelligence (AI) and its remarkable results in different fields of science, many models have also been proposed for malware detection. AI models enable algorithms to operate autonomously without explicit programming. AI models are fed with new data, and they can independently learn, grow, develop, and adapt [4]. Researchers have been trying to develop effective and efficient techniques to detect and classify current and new malware using minimal resources. The ability of AI models to learn, grow, develop and adapt can be integrated into malware analysis, addressing the problem of dealing with the detection of malwares. **Table 1** list the properties of different analysis approach.

Static Analysis	Dynamic Analysis	AI Analysis
● Fast. ● Can only detect the known malwares. ● Fails to detect obfuscated and new variants of malware.	● Slow. ● Can detect known and new types of malwares. ● Fails to detect malwares equipped with time delays and user interaction.	● Fast. ● Can be used to detect known and new malwares. ● Can be used to detect malwares equipped with different techniques.

Table 1: Properties of different malware analysis technique

Microsoft's Windows is the most widely used computer operating system in the world, accounting for a 70.68% share of the desktop, tablet, and console OS market in August 2022 [5]. This survey aims to review and analyze a few of the latest literature that utilizes AI models for detecting and identifying malware designed for Windows Operating systems (OS). This paper will be reviewing the following points in the papers:

1. The problem addressed.
2. The features extracted from the Portable Executable (PE).
3. The Artificial Intelligence (AI) model used.
4. The dataset used for training the model.
5. The result of the proposed model.

Literature Review:

Reference [6] used the kNN model and found the optimum value of k for effective malware classification. The dataset used contains 43,876 samples. Each sample consisted of three features: MD5 hash, API call type, and class label. The highest accuracy of 98.17% was achieved using the value of k=3. The value other than 3 resulted in less accuracy. The author didn't specify the false positive and negative rates of the model. The dataset used for training contained shallow features that can be manipulated using obfuscation and junk code insertion techniques to misclassify a malware as a benign file.

Reference [7] focused on decreasing the false positive rate to zero. It stated that the data the model use during the training phase for learning is very crucial because if model learns a wrong feature, it will generate false results which may have serious consequences in the real world. Secondly, it stated that while using a deep learning model, the operation inside models are non-understandable by humans, and in case of a false alarm, it is difficult to understand whether it was data or the model itself. The authors used five different ML models (Random Forest, AdaBoost, Gradient Boosting, Decision Trees, and Gaussian Naive Bayes) to train on a dataset whose size was 50,000 and had 50 independent variables and 1 dependent variable. The training dataset size was set to 80% and the testing dataset size was set to 20%. The variables were featured extracted from the binary file header. In the results, the random forest had the highest accuracy while Gaussian Naive Bayes had the lowest. The random forest model also achieved a 0.48992% false positive rate and 1.008782% false negative rate, the false positive rate was not zero but it was near. The paper was near to achieving the stated goal of zero false positive rate. It compared different ML model accuracy while training on the same dataset concluding that choice of model is crucial in efficient malware detection.

Reference [8] proposed the two-stage LSTM model using the opcode sequence of the file to learn automatically whether it is malware or not. The sample files were disassembled using IDA pro and then input to the algorithm to extract the opcode sequences from each sample file. The process of extracting opcode sequences was automated. The dataset used for training and testing purposes consisted of 969 malwares and 123 benign files collected from Microsoft, MalwareDB, and VirusShare. The model was tested on different word embedding techniques and word windows sizes. It achieved the highest accuracy of 97.87% for binary classification and 94.51% for multi-classification with CBOW word embedding and 10-word window size. The dataset only consisted of opcodes, if combined with operands, the model may have resulted in higher accuracy along with more enriched information as input to the model.

Instead of utilizing high-level features, the reference [9] took a step further in utilizing low-level features for malware classification and detecting obfuscated malwares. This technique allowed capture of information at the micro-architecture level because obfuscated malwares can’t alter the final data on the General Purpose Registers (GPRs). The spatial and temporal properties in GPRs were used as features to the model. The authors also reduced the size of the GPRs instructions to 0.8K instructions per sample in the dataset as compared to the previously proposed models requiring 25K to 10K instructions per sample. The proposed model “FusionST” has four CNNs that abstracts the spatial correlation between GPRs followed by LTSM blocks encoding the sequential information for final malware detection. The dataset used for training and testing was collected from VxHeaven (60%) and VirusShare (40%). The purposed model achieved an accuracy of 95.8% and a FPR of 0.068. The dataset has to be converted into model readable form by executing samples in the controlled environment to extract relevant features, which is a time-consuming process and is not ideal for end-user devices.

Reference [10] focused on detecting the general types of malware instead of targeting specific malwares. It used a custom-designed convolutional neural network (CNN). A custom CNN and pre-trained models (AlexNet, ResNet, VGG, and Inception V3) were used for training. The input to the model were 2D Grayscale image of malwares. The mallMg dataset having 9458 samples and 25 different classes was used. The training dataset size was set to 70% and the testing dataset size was set to 30%. The pre-trained model Inception-V3 had the highest accuracy of 98.90%. The authors utilized the pre-trained models for training the malware classifier with over 25 different classes of malware achieving a better result than custom CNN.

Reference [11] treated malware detection as a graph classification problem. The custom Graph kernel-based Support Vector Machine (SVM) model was used for malware detection. The API call graph which was input to the model was constructed for each sample by disassembling it and then extracting the Control Flow Graph (CFG). The API call graph size was reduced by merging the vertices of API calls that correspond to the same API function, the resulting graph was named the abstract API call graph. This graph was constructed only using static analysis. The model was trained and tested on a dataset containing 6291 samples collected from VxHeavens and 2323 benign samples from Windows Application and CygWin. The proposed model had 98.93% accuracy and a 1.24% False Positive Rate (FPR). The model was also compared again famous AntiViruses by generating 180 new malwares, it had a 100% detection rate followed by McAfee which had a 96% detection rate. The author in this paper constructed a graph of process behavior statically which was enrich in information to give clear patterns and data to the model to detect malware and it gave very promising results when the model was compared with famous Antiviruses.

Reference [12] collected short sequences of data during initial stage of execution. Each sample was executed for 20 seconds while taking snapshots of metrics every second. As recent works have shown that API calls are vulnerable to manipulation ([13], [14] citied in [12]). The proposed model used machine activity instead of API calls as a feature.

The machine activity metrics allow the model to infer information from the completely unseen input data. The data was collected by executing binary files in the Cuckoo Sandbox. The authors used the recurrent neural network (RNN) model for training. The dataset was composed of 2,345 benign and 2,286 malicious files, obtained from the VirusShare and VirusTotal. The dataset was split into training and testing on the basis of creation time of malware. So, while testing, the model is exposed to a completely new variant of malwares. The proposed model achieved an accuracy of 96.01% and a FPR of 3.17%. It also stated that models need to be trained from time to time to tackle the challenge of detecting the new variant of malware every day. The model has a high FPR, and requires execution of the file for specific time which can be bypassed by time delay evasion technique.

The reference [15] used the Siamese Neural Network (SNN) utilizing the entropy feature instead of the image feature to train the model for ransomware classification. The proposed model was trained on the pre-trained model VGG-16. The authors stated the using entropy feature over the image feature had an advantage because image features can be easily influenced by the white noise and spatial feature while entropy is less sensitive to small changes made to the generic code([16] [17] [18] citied in [15]). The aim of the proposed model was to detect ransomware by training on a few samples for a class. The dataset was composed of 1046 samples belonging to 11 different classes obtained from the Virus Share. The model had a 98.2% accuracy and 80.3% precision. The model requires longer training time because it needs to train a high number of parameters and it can only detect ransomware belonging to a family explored in the training dataset which makes it not usable in the fast-changing environment.

Reference [19] proposed an automated unsupervised malware detector. The framework consisted of two models: the kMean model for creating clusters and labeling the samples then, a deep neural network embedded with a feature attention block for training and testing. The dataset compromising of 15,457 samples was collected over a period of seven months using honeypots. The data was labeled and then validated using VirusTotal web API for validation. The framework achieved an accuracy of 98.09% and a False Positive Rate of 2%. The model only used pattern-based features as input to the model. The result of the unsupervised approach to detecting malware resulted in good accuracy and with improvements, it can be used to achieve the more promising result.

Reference [20] focused on extracting the structural information from the executable in the form of a hierarchical graph. The hierarchical graph incorporates the inter-function call graph with the intra-function call graph. The dataset was composed of 227,197 samples obtained from the VirusTotal. The samples were unpacked using unpacker tools and the threshold of file size was 2MB. The inter-function call graph nodes were limited to 3,000 and intra-function call graph nodes were limited to 10,000 reducing the total number of samples to 183,028. Each sample was disassembled using IDA Pro and a hierarchical graph was constructed. The proposed model MalGraph achieved an accuracy of 99.97% and a false-positive rate (FPR) of 0.1%. The robustness of the proposed malware was also evaluated by selecting 1,000 of the correctly classified samples and MLSEC-Attack was performed using these samples until a successful adversarial malware was generated. Among all the baseline methods, the MalGraph outperformed with exceptional results. Unlike baseline methods, which either extract pattern-based features or manually crafted features. The proposed framework focused on extracting the features that represent the behavior of the executable. Training the model on the features extracted by the flow of external function along with the flow of local function helped to achieve promising and exceptional results.

Reflection & Analysis:

The choice of the features, dataset, model, and efficient combination of these altogether play a crucial role in developing an efficient framework for malware detection.

The review of the literature revealed that most of the models are targeting shallow features obtained from static analysis such as MD5 hash, DllCharacteristics, and machine in binary files, which don't contain any behavioral information and are solely pattern-based information. These pattern-based features are very generic and may result in a high rate of misclassification of samples outside the specific dataset.

The others targeting more specific features required execution of the sample in a controlled environment which can be bypassed by various techniques embedded in malware such as User-interaction, time-based methods, and environmental awareness.

Reference [10] used the 2D grayscale image constructed from the raw bytes of the binary files. The constructed images are of different sizes and shapes and need to scale up and down to specific shapes and sizes resulting in a loss of information. The obfuscation technique may generate a malware that is unique and new to the model and may result in misclassification.

Reference [20] addressed the majority of the issues by constructing a behavioral-based graph using static analysis. However, the dataset required for training the model was very large. Certain limitations were also imposed while extracting the features from sample files. This leaves a major gap in detecting newly emerging variants of the malware which have no or few samples available. Over time, certain few-shot meta-learning models have been proposed to detect malware by training on pre-trained models and small datasets. However, the accuracy is not up to the desired mark.

Misclassifying a single benign file in millions as malware can result in serious problems for the user. This metric is measured by False Positive Rate (FPR) and holds great importance that was not discussed and emphasized in the majority of the papers. Most of the AI models are black-box, which means that in case of false alarms, it is difficult to predict whether it was data or the model.

The proposed models have tried to address specific issues associated with AI-based malware detection such as using more rich information as input, using pre-trained models, and reducing the dataset size. Training a model on large amount of dataset needs high computational resources and time which makes the currently proposed papers impractical in the real world.

Future work needs to be focused on reducing the training dataset. Input features needs to be consisting of rich behavioral and structural information of the sample. The models should also focus on keeping the False-Positive rate as low as possible.

Conclusions:

Due to the significant growth of malware, traditional techniques for malware detection are less effective now. The Anti-Viruses solutions still rely on signature-based and behavior-based detection. Malware detection demands an intelligent solution now more than ever. Although many AI models have been proposed with excellent results for malware detection. This survey reviewed some of the latest proposed models for malware detection. The survey revealed that the currently proposed models in AI are impractical in the real world due to the requirement of large datasets for training and targeting shallow features as input to the models from the input. They are also unable to keep up with the evolving nature of the malware and need to be trained from time to time on new samples of the malware.

References

[1] K. Bober, "2022 Must-Know Cyber Attack Statistics and Trends," 27 October 2022. [Online]. Available: https://www.embroker.com/blog/cyber-attack-statistics/#:~:text=Cybercrime%2C%20which%20includes%20everything%20from,of%20the%20COVID-19%20pandemic..

[2] AV-ATLAS, "Malware," AVTEST, 2022. [Online]. Available: https://www.av-test.org/en/statistics/malware/.

[3] SONICWALL, "CYBER THREAT INTELLIGENCE FOR NAVIGATING THE UNKNOWNS OF TOMORROW," SONICWALL, California, 2022.

[4] V. Kanade, "What Is Machine Learning? Definition, Types, Applications, and Trends for 2022," 30 August 2022. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/.

[5] S. R. Department, "Market share held by the leading computer (desktop/tablet/console) operating systems worldwide from January 2012 to August 2022," 29 August 2022. [Online]. Available: https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/#statisticContainer.

[6] T. A. Assegie, "An Optimized KNN Model for Signature-Based Malware Detection," *International Journal of Computer Engineering In Research Trends (IJCERT)* , vol. 8, no. 02, p. 4, 2021.

[7] A. Dewanje and K. A. Kumar, "A New Malware Detection Model using Emerging Machine Learning Algorithms," *I.J. of Electronics and Information Engineering, Vol.13, No.1, PP.24-32*, p. 9, 2020.

[8] R. Lu, "Malware Detection with LSTM using Opcode Language," p. 7, 2019.

[9] F. Li, C. Yan, Z. Zhu and D. Meng, "A Deep Malware Detection Method Based on General-Purpose Register Features," *International Conference on Computational Science*, vol. 11538, p. 15, 2019.

[10] M. Alam, A. Akram, T. Saeed and S. Arshad, "DeepMalware: A Deep Learning based Malware Images Classification," in *2021 International Conference on Cyber Warfare and Security (ICCWS)*, 2021.

[11] K.-H.-T. Dam and T. Touili, "Malware Detection based on Graph Classification," in *3rd International Conference on Information Systems Security and Privacy*, 2017.

[12] M. Rhode, P. Burnap and K. Jones, "Early Stage Malware Prediction Using Recurrent Neural Networks," *Computers & Security*, vol. 77, pp. 578-594, 2018.

[13] I. Rosenberg and E. Gudes, "Bypassing system calls-based intrusion detection systems," *Concurrency and Computation: Practice and Experience*, vol. 29, 2016.

[14] I. Rosenberg, A. Shabtai, L. Rokach and Y. Elovici, "Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers," in *RAID 2018*, 2018.

[15] J. Zhu, J. Jang-Jaccard, A. Singh, . I. Welch, H. AI-Sahaf and . S. Camtepe, "A Few-Shot Meta-Learning based Siamese Neural Network using Entropy Features for Ransomware Classification," 2022.

[16] T. Hamid, D. A.-J. Obe and J. Mustafina, "Evaluation of the Dynamic Cybersecurity Risk Using the Entropy Weight Method," *Technology for Smart Futures*, pp. 271-287, 2018.

[17] J. Zhu, J. Jang-Jaccard and P. A. Watters, "Multi-Loss Siamese Neural Network With Batch Normalization Layer for Malware Detection," *IEEE Access*, vol. 8, pp. 171542-171550, 2020.

[18] J. Zhu, J. Jang-Jaccard, A. Singh, P. A. Watters and S. Camtepe, "Task-Aware Meta Learning-based Siamese Neural Network for Classifying Obfuscated Malware," 2021.

[19] S. K. J. Rizvi, W. Aslam, M. Shahzad and S. Saleem, "PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable," *Complex & Intelligent Systems,* vol. 8, pp. 1-13, 2021.

[20] X. Ling, L. Wu, W. Deng, Z. Qu, J. Zhang, S. Zhang, T. Ma, B. Wang, C. Wu and S. Ji, "MalGraph: Hierarchical Graph Neural Networks for Robust Windows Malware Detection," in *IEEE Conference on Computer Communications*, 2022.