Grenze International Journal of Engineering and Technology, June Issue

Developing a Automation Malware Detection Tool using Machine Learning Algorithm

S Tamil Selvi[1], Nisha A[2], Dharun Kumar M[3] and Hemanth TS[4]
[1-4]Department of Cyber Security, SRM Valliammai Engineering College
Email: tamilselvy@gmail.com, nishaanandan123@gmail.com,dharun0473@gmail.com,eswarhemanth82@gmail.com

***Abstract*— A instructive cybersecurity concern is the increasing sophistication of malware, with Portable Executable (PE) files acting as an ordinary channel for harming activity. This paper presents a machine learning-based method for PE malware detection in response to the ever-expanding threat landscape. Using a large benchmark dataset, we carefully evaluate the properties of PE files using the Hyperparameter Tuned KNN (HTKNN) machine learning method to improve malware detection precision. Grid Search Cross-Validation (CV) is used to optimize performance and fine-tune the model parameters. Our work offers a reliable and efficient method for identifying and reducing malware dangers encoded in PE files, which supports the continuous efforts to strengthen cybersecurity defenses. The outcomes of our assessment emphasize. The quality of the feature representation, the quantity and variety of the dataset, and the distance metric selected all affect how successful the K-Nearest Neighbors (KNN) algorithm is at detecting malware. KNN is a straightforward yet effective technique that uses the majority class among its k nearest neighbors in feature space to classify data items.KNN's simplicity and ease of implementation are two of its benefits. Because it is non-parametric, it does not assume anything about the data's underlying distribution. Because of its adaptability, KNN can record intricate decision boundaries, which makes it useful for identifying both known and unknown viruses.**

***Index Terms*— Malware, Blockchain, KNN, Threats.**

I. INTRODUCTION

Computers, cellphones, networks, cloud servers, and other digital systems can become infected with malicious software, or malware, which can cause harm to the systems or users. The extent of injury to a system can range from decreased functionality to non-operation or theft of confidential data; the extent of harm to users can vary from destruction of their files to identity theft when utilizing a compromised system. Malware is categorized as adware, backdoor, bot, downloader, launcher, ransomware, rootkit, spyware, Trojan, virus, worm, etc. based on how it behaves and creates damage to an infected system. Detecting distinctive signature strings allowed researchers to identify early malware, also referred to as classical malware. Nevertheless, a race between those who create malware and those who create anti-malware software soon began. More recent malware use clever and advanced methods. Numerous techniques were created for the examination and mining of sophisticated malware in order to identify its fundamental traits. These mining programs examined DLL function calls, assembly instructions, hexadecimal sequences, PE file headers, and/or combinations of these characteristics in addition to malware signatures. Before feeding the characteristics into a tool, they can be subjected to additional processing

Grenze ID: 01.GIJET.10.2.200

after the initial pass. For instance, hashing techniques and entropy are applied. Be aware that the so-called "Hashing Trick" is a tried-and-true dimension reduction method with a strong theoretical base, not a gimmick. Advanced malware is tenacious and adaptive, and once it enters a system, it escapes detection.Therefore, software files should be screened for malignancy before being added to a digital system. Methods of evaluating software are classified as static, dynamic, and hybrid .Software is analyzed dynamically by running it in a "sandbox," which is a controlled environment designed to resemble end-user operating systems, and watching and documenting its actions. The software's recorded behavior is assessed manually or using machine learning techniques to determine the final classification. Most of the time, dynamic analysis is not viable due to its length, cost, and time commitment.Software features are extracted from the program by analysis or mining, and the program is then classified using the features that were extracted. This process is known as static analysis of software. The characteristics required for categorization vary depending on the instrument employed.Heuristic-based,model-checking-based,behavior-based, signature-based, and machine learning-based are the five general types of static analysis tools. The code of the malware is quickly altered to avoid detection by the creators, who have access to all of the tools for this purpose.

II. Literature Survey

Ö. Aslan and A. A. Yilmaz [1] discussed the shift in human activities from physical to virtual realms, accelerated by recent advancements in computer systems and further propelled by the COVID-19 pandemic. Cybercriminals are increasingly targeting virtual vulnerabilities using evolving malware. Researchers used datasets like Malimg, Microsoft BIG 2015, and Malevis to develop a highly accurate malware categorization approach, outperforming many current methods. Their solution achieved 97.78% accuracy on the Malimg dataset.

[2] outlines a novel approach developed by D. Javaheri, P. Lalbakhsh, and M. Hosseinzadeh, introducing an innovative solution to identify elusive and shape-shifting malware programs, addressing the challenge of limited datasets for modelling these types of malwares. Researchers used genetic algorithms and optimization techniques to generate mutated elite malware samples, addressing the lack of training data. This method significantly boosted detection accuracy, especially for uncommon and shape-shifting malware, showcasing effectiveness in tackling elusive threats.

Darem A.A., Ghaleb F.A., Al-Hashmi A.A., Abawajy J.H., Alanazi S.M. , Al-Rezami A.Y. [3] presented a solution addressing the challenges posed by evolving malware variants, which represent significant and ever-changing threats to cybersecurity and have the potential to cause substantial damage to computer systems.Traditional malware detection struggles with the dynamic nature of malware variants, leading to outdated models. This demonstrates the effectiveness of their strategy against evolving malware.

The various capabilities and combination of man- made reasoning (man-made intelligence) applications in the Android working framework, which have created its dominant position in the market, are discussed by Singh J., Thakur D., Gera T., Shah B., Abuhmed T, Ali F. [4]. The feature Fusion-SVM model achieved a high accuracy rate of 93.24% in detecting and classifying complex Android malware using malicious pictures and certificate formats, showcasing its effectiveness against advanced threats.

Cilleruelo C., Enrique-Larriba, and et al[5] address the persistent challenge of Potentially Harmful Apps (PHAs), which pose a threat similar to conventional malware. Researchers propose an innovative machine learning solution to detect Potentially Harmful Apps (PHAs) in the Google Play Store. Their method achieved a high 90% accuracy rate when tested on a dataset of 91,203 apps. This approach, based on lifespan data, provides a new dimension in the ongoing battle against PHAs in app ecosystems and can complement existing detection models.

A novel approach for automated Android malware detection was put out by brahim, B. Issa, and M. B. Jasser [6]. Researchers conducted experiments focusing on Android malware detection and classification. Their model outperformed existing methods, achieving a remarkable F1-score of 99.5% for two-class identification and 97% across all five classes. These results highlight the effectiveness of their approach in accurately detecting and classifying Android malware.

A.Al-Hashmi et al.'s [7] novel method for identifying malware programs, many of which are variants of already-existing malware such polymorphic and transformative malware, is introduced. Malware variants often change behavior to evade security measures, making identification challenging. Researchers incorporated these features into an outrageous angle helping based classifier for independent navigation. Evaluations revealed their model increased detection rates and reduced false negatives, making it reliable for malware variant identification.

A solution for spotting malware behavior is put out by Nguyen H.N. , Abri F., and et al[8]. Static and dynamic exams are the two main approaches used to analyze malicious apps. Static analysis involves parsing a malicious program without executing it,found that utilizing XOR to obfuscate code was the most efficient method. Code

obfuscation allows for the creation of malware variants that may avoid detection by an average of 28 distinct malware engines. These discoveries can be useful in creating anti-malware programs that are capable of spotting newly discovered malware variants.

He t., Han C., Isawa R. and et al[13] discuss the enormous social difficulties brought on by the quick rise of recently developed forms of Internet of Things and its variations. Researchers propose an innovative online processing approach for building phylogenetic trees from large malware datasets, addressing time constraints. Using 65,494 IoT malware samples, their method reduced computing time by 97.52% compared to conventional methods. The clustering approach accurately classified family names with a 95.5% accuracy rate and pattern names with a 99.3% accuracy rate. This advancement is crucial for analyzing and classifying massive IoT malware datasets efficiently generating artifacts like control-flow graphs. Dynamic analysis, in contrast, requires running the program. The technique introduces MalView, a platform for malware investigation using intelligent visualization. MalView supports behavior analysis and signature-based artifacts, aiding tasks such as characterizing malware, locating payloads, analyzing unknown malware, and examining affected system components.

The work of Almomani I., Alkhayer A., El-Shafai W.[9] on establishing financially savvy profound learning (DL)-based calculations for distinguishing Android malware (AMD) frameworks. Technique introduces a vision-based Android malware detection (AMD) model utilizing sixteen optimized convolutional neural networks (CNNs). Compared to traditional vision-based algorithms, this CNN-based AMD model outperforms previous techniques on benchmark Android datasets.

Pattee J., Anik S.M. , and Lee B.K. [10] propose hardware solutions for detecting malware, a method gaining prominence due to the susceptibility of software-based solutions to intelligent malware compromise. To enhance real-time malware detection accuracy, they propose schemes to improve accuracy, such as introducing additional PMCs, alongside hardware acceleration methods. These combined strategies result in accuracy improvements (5-10%) and faster detection speeds (up to 10%), with a negligible impact on hardware complexity (less than 1% of chip complexity).

The efficacy of several AI (ML) models for malware discovery is explored by H. Manthena, Kimmel J.C. , Abdelsalam M., and Gupta M. [11]. The technique helps to evaluate various ML models, including SVM Direct, SVM-RBF, Irregular Woods, Feed-Forward Brain Organizations, and Convolutional Brain Organizations, for internet spyware identification. To enhance interpretability, the Shapley Additive Explanations (SHAP) method was employed, utilizing different approaches like KernelSHAP, TreeSHAP, and DeepSHAP. The aim is to make ML models' decisions, particularly in online malware identification.

Jin, Choi J., Hong J.B. , Kim H. [12], emphasises production of malware variations utilizing different evasion techniques to bypass malware detectors. Understanding the specific properties that drive these variations to evade malware detection methods is crucial. The researchers

Convolutional Neural Networks (CNNs) are successful in classifying and detecting malware based on perception, according to Belal M.M., Sundaram

D.M. [14]. The technique introduces the Butterfly Vision Transformer, a novel structure of perspective- based spyware characterization, identification. By capturing both nearby and worldwide spatial portrayals of malware pictures, B_ViT outperforms other Vision Transformer variants, achieving accelerations of 2.42 times over IEViT and 1.81 times over typical ViT variants. The study highlights B_ViT's effectiveness in detecting surface-based malware and its adaptability to handle polymorphic obfuscation methods, surpassing CNN-based malware classification techniques.

The vital need to safeguard all layers and devices inside a computing environment, especially under edge/haze computing situations, is discussed by Gulatas I., H. H. Kilinc, ZaimA.H., Aydin M.A. [15]. IoT devices, vulnerable components in secure data environments, face threats from emerging Edge/Haze computing-based malware. Researchers used analytic methodologies like "Digital Kill Chain" and "Miter ATT&CK for ICS" to characterize these malware families, highlighting their unique characteristics. The study emphasizes the importance of understanding and addressing the challenges posed by Edge/Haze computing malware, particularly in the context of IoT devices.

A. Inference

[1] Leading to an increased focus on cybersecurity as researchers develop highly accurate malwarecategorization methods. [2] Genetic algorithms are utilized to generate mutated malware samples.

[3] A dynamic malware detection model surpassing traditional methods, new infections with minimal monthly updates [4] The detecting and classifying complex Android threats, demonstrating its efficacy against advanced malware varieties. [5] Detection of Potentially Harmful Apps (PHAs), offering a lifespan-based approach to

enhance the fight against PHAs. [6] an automated Android malware detection method surpassing existing approaches, in accurate malware detection and classification. [7] A novel method incorporating outrageous angle-based classification for identifying malware variants. [8] Facilitating behavior analysis and signature-based artifact examination for malware investigation.
[9] Develop a powerful yet efficient deep learning model for Android malware detection. [10] Proposed hardware solutions for malware detection, enhancing accuracy and speed through additional PMCs. [11] Evaluate AI models with SHAP to enhance transparency in online malware identification. [12] A framework utilizing XOR obfuscation to create malware variants evading detection for anti-malware development. [13] Proposed an online processing approach for efficiently analyzing large IoT malware datasets.
[14] Outperforming CNNs in malware classification by capturing spatial representations and achieving faster detection. [15] Safeguarding IoT devices from Edge/Haze computing-based malware, unique traits and addressing emerging threats categorization methods. [2] Genetic algorithms are utilized to generate mutated malware samples. [3] A dynamic malware detection model surpassing traditional methods, new infections with minimal monthly updates.

A. Data collection

Data scientists and machine learning aficionados frequently use Kaggle, a prominent platform where they can locate datasets, take part in contests, work together, and pick up tips and ideas. Finding relevant datasets for your study or project on Kaggle, downloading them, and then utilizing tools like Python and R to explore and analyze them are the usual steps in the data collection process. Kaggle is an excellent tool for gathering data in a variety of industries because it offers a large range of datasets on subjects including sports, entertainment, healthcare, and finance. Furthermore, organized datasets are frequently offered by Kaggle, requiring little preparation before analysis can begin.In order to produce insights or facilitate decision-making processes, data collection entails obtaining information from a variety of sources. Techniques like surveys, interviews, observations, and dataset analysis might be a part of it. Planning, creating tools for gathering data, gathering the information, organizing and cleaning it, and then analyzing it to make inferences are usually steps in the process. When handling personal data, it is imperative to adhere to privacy standards and make sure the data is accurate, relevant, and ethically sourced.threats, highlighting the importance of vigilant cybersecurity practices.

```
array([[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
        -1.27388769e-01, -1.53846154e-01,  7.37280000e+04],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
         8.25532773e-01,  4.73372781e-01,  0.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
        -2.07974177e-01, -2.60355030e-01,  0.00000000e+00],
       ...,
       [ 0.00000000e+00, -1.44000000e+02, -3.00000000e+00, ...,
        -3.76994461e-01, -2.01183432e-01,  0.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
         5.81374023e-02, -8.28402367e-02,  0.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
         2.12553644e+00,  1.23076923e+00,  0.00000000e+00]])
```

B. *Data Cleaning*

This process entails finding and fixing data mistakes or discrepancies. Typical duties consist of: Managing absent values: Choose between using sophisticated methods like interpolation, impute missing values, or remove rows and/or columns with missing data. Eliminating duplicates: To guarantee data integrity, locate duplicate records and delete them. Fixing mistakes: correcting typos, inconsistent formatting, and other data errors.

C. *DATA SCALING*

The goal of data scaling, a basic preprocessing method in machine learning, is to bring a dataset's features to a common scale without changing the data's underlying distribution. For algorithms that are sensitive to feature scale, this normalization or standardization phase is essential to ensuring that every feature contributes equally to the model's learning process. Standardization, which changes characteristics to have a mean of 0 and a standard deviation of 1, and Min-Max Scaling, which rescales features to a given range usually between 0 and 1, are common scaling techniques. While normalization guarantees that every sample has a unit norm, which is advantageous for algorithms depending on distance measurements, robust scaling, which uses the median and interquartile range, is resistant to outliers.

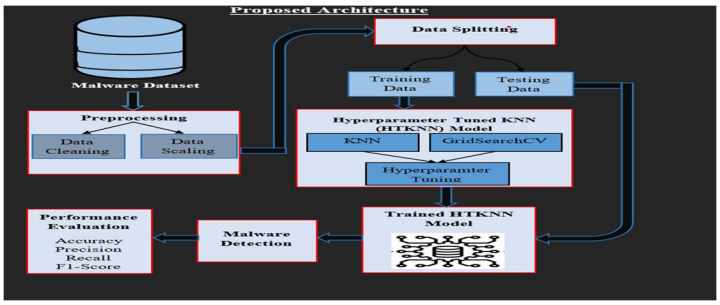


Figure.1 Working model of automation malware detection

III. Model Building And Training

Two subsets of the preprocessed dataset are created: a test set and a training set. This separation is necessary to train the machine learning model on one set of data and assess its output on an other, hidden set. The model can identify patterns and relationships in the training set, which usually contains most of the data. It may also evaluate its generalization and prediction skills using the test set, which is a separate dataset.A variation of the K-Nearest Neighbor technique, the Hyperparameter Tuned K-Nearest Neighbor (HTKNN) classifier aims to improve performance by using adjusted hyperparameters. The K-Nearest Neighbor algorithm is configured with specific hyperparameters that affect how it behaves during the learning process in order to apply it. To improve accuracy and prevent overfitting or underfitting problems, this tweaking is essential.

A. MODEL TESTING

The next critical step is to evaluate the Hyperparameter Tuned K-Nearest Neighbor (HTKNN) model's performance on a different, previously unreleased test dataset, following training it on the selected training set. This assessment is crucial for ascertaining the model's effectiveness in the actual world since it sheds light on the model's capacity to apply newly discovered patterns to previously undiscovered data. The following specific steps are involved in the assessment process:

Test Dataset Application: For this assessment, the preprocessed dataset—which had previously been divided into training and test sets—is used. Different from the training data, the test dataset simulates the model's performance in an actual environment by representing a scenario where the model comes across new instances.

Model Prediction: Using the characteristics found in the test samples, the trained HTKNN model is applied to the test dataset to generate predictions. The model classifies each instance as either benign or malicious by using the patterns it learned during the training phase.

B. MODEL EVALUATION

Using important metrics, the performance of the proposed Hyperparameter Tuned K-Nearest Neighbor (HTKNN) model is thoroughly assessed to give a detailed understanding of its efficacy in malware detection. The computation and interpretation of accuracy, precision, recall, and F1-score are described in detail below:

Accuracy: By dividing the number of correctly classified examples by the total number of instances, accuracy determines how accurate the model's predictions are overall.

Precision: Precision calculates the ratio of true positive predictions to all projected positive cases in order to evaluate the accuracy of positive forecasts.

Sensitivity or true positive rate is another name for recall, which assesses how well the model captures all real positive examples. It is the proportion of all actual positive occurrences to all true positive expectations.

The F1-score is a balanced statistic that takes into account both false positives and false negatives. It is calculated as the harmonic mean of precision and recall.

A model that performs well in terms of overall forecast correctness is indicated by a high accuracy. The precision of a model is its capacity to reduce false positives, or the likelihood that cases that are flagged as positive are actually true positives. Recall shows that the model can reduce false negatives by capturing a sizable percentage of true positive cases. The F1-score offers a fair evaluation and is especially helpful in situations where recall and precision must be taken into account simultaneously.

A high accuracy indicates a well-performing model in overall prediction correctness. Precision reflects the model's ability to minimize false positives, ensuring that instances classified as positive are indeed true positives. Recall demonstrates the model's capability to capture a significant proportion of actual positive instances, reducing false

negatives. F1-score provides a balanced assessment, particularly valuable in scenarios where precision and recall need to be considered together.

C. WEB APPLICATION

Interface for Input: The web application has an easy-to-use interface for users to upload or enter Portable Executable (PE) files to be checked for malware. A file upload button might be present in the interface, enabling users to choose files straight from their devices.

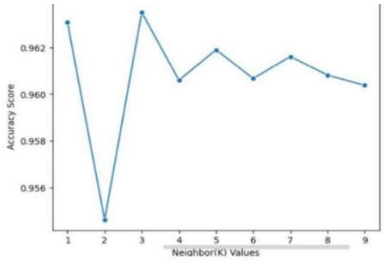
The application does pre-processing, such as data cleansing and scaling, on the data after the user uploads or inputs the PE files. By doing this, the input data is guaranteed to be in a format that the HTKNN model can use.

Utilizing frameworks like Flask, Django, or Node.js, develop the web application. To enable prediction-making, incorporate the machine learning models into the application. To make sure the web application functions as intended, extensively test it. Analyze its performance using task-relevant metrics, like F1-score, recall, accuracy, and precision. Create a web application with an intuitive user interface that makes it simple for users to interact with the automation detection tool. Think of elements like user feedback and data presentation. To provide user access, deploy the web application to a server or cloud platform. Make certain that the deployment infrastructure is scalable and reliable. Keep an eye on the web application's and the machine learning models' performance all the time. When necessary, update the models with fresh

IV. COMPARISON OF EXISTING AND PROPOSED MODEL

TABLE I– COMPARISON OF EXISTING AND PROPOSED SYSTEM

S.NO.	ALGORITHM	ACCURACY
1.	[1] Grid Search CVV	91.56%
2.	**[**9] Convolutional Neural Networks (CNN)	93.24%
3.	[11] Hyper tuned (HTKNN)	90.36%
4.	K-nearest neighbour (knn)	90%



Graph.1 - Comparison graph of existing and proposed system

VII. CONCLUSION

To sum up, our research highlights the urgent need for novel approaches to combat the growing threat of sophisticated malware,especially that which infects Portable Executable (PE) files. With the use of the Hyperparameter Tuned KNN (HTKNN) technique and a machine learning approach, we have shown how to detect harmful content hidden in PE files in a reliable and efficient manner. Grid Search Cross-Validation (CV) has improved the performance of our model and ensures higher malware detection accuracy. Based on an extensive benchmark dataset, our results confirm the effectiveness of the suggested methodology and show how much it may add to the continuing efforts to strengthen cybersecurity defenses.

FUTURE ENHANCEMENT

To take use of deep learning architectures' ability to identify complex patterns in the PE files, think about putting neural networks or deep neural networks into practice. For this, one may investigate recurrent neural networks (RNNs) or convolutional neural networks (CNNs). Create and include sophisticated machine learning models for anomaly detection in order to nimbly recognize novel automation trends.Use reinforcement learning techniques so that the tool can keep becoming better at detecting things based on changing automation. Improve the efficiency of data processing and minimize latency to optimize the tool for real-time monitoring.Use predictive analytics to anticipate possible automation tasks and proactively stop dangerous or unlawful actions. Increase the tool's capacity to identify a variety of automation methods, such as AI-driven automation, robotic process automation (RPA), and script-based automation.Investigate and include techniques for identifying difficult automation scenarios, like emulating human behavior or combining several automation kinds.To improve the detection tool's accuracy and reduce false positives and negatives, thoroughly test and validate it. Put feedback systems in place so that the tool's algorithms can gradually be improved by learning from user input. Create the solution with

seamless scalability in mind to accommodate growing user bases and data volumes, guaranteeing dependable performance in bigger enterprise environments.Investigate cloud-based options to help scalability and economical use of resources.Keep up with current developments in the automation space and make sure the product is flexible enough to accommodate different approaches and platforms.Work together with additional cybersecurity tools and technologies to offer all-encompassing defense against changing threats.Make the tool's user interface easier to understand and use so that security experts can quickly assess and take appropriate action based on the information provided. Give users the ability to customize dashboards and reporting elements to meet varying corporate requirements.Use behavioral analysis approaches to comprehend typical automation practices in certain situations so that the tool can detect anomalies effectively.Develop mechanisms to distinguish between authorized and unauthorized automation activities based on behavioral patterns. Integrate the automation detection tool with incident response systems to automate responses or trigger alerts for security teams. Provide detailed forensic data and analysis capabilities to investigate and respond to automation-related incidents effectively. Ensure the tool complies with relevant data protection and privacy regulations, providing features for audit trails and reporting to support compliance efforts. Regularly update the tool to align with changing regulatory requirements in the cybersecurity land

REFERENCES

[1] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[2] D. Javaheri, P. Lalbakhsh and M. Hosseinzadeh, "A Novel Method for Detecting Future Generations of Targeted and Metamorphic Malware Based on Genetic Algorithm," in IEEE Access, vol. 9, pp. 69951- 69970, 2021, doi: 10.1109/ACCESS.2021.3077295.

[3] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi and A. Y. Al-Rezami, "An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning," in IEEE Access, vol. 9, pp. 97180-97196, 2021, doi: 10.1109/ACCESS.2021.3093366.

[4] J. Singh, D. Thakur, T. Gera, B. Shah, T. Abuhmed and F. Ali, "Classification and Analysis of Android Malware Images Using Feature Fusion Technique," in IEEE Access, vol. 9, pp. 90102-90117, 2021, doi: 10.1109/ACCESS.2021.3090998.

[5] C. Cilleruelo, Enrique-Larriba, L. De-Marcos and J. -J. Martinez-Herráiz, "Malware Detection Inside App Stores Based on Lifespan Measurements," in IEEE Access, vol. 9, pp. 119967-119976, 2021, doi: 10.1109/ACCESS.2021.3107903.

[6] M. İbrahim, B. Issa and M. B. Jasser, "A Method for Automatic Android Malware Detection Based on Static Analysis and Deep Learning," in IEEE Access, vol. 10, pp. 117334-117352, 2022, doi: 10.1109/ACCESS.2022.3219047.

[7] A. A. Al-Hashmi et al., "Deep-Ensemble and Multifaceted Behavioral Malware Variant Detection Model," in IEEE Access, vol. 10, pp. 42762-42777, 2022, doi: 10.1109/ACCESS.2022.3168794.

[8] H. N. Nguyen, F. Abri, V. Pham, M. Chatterjee, A. S. Namin and T. Dang, "MalView: Interactive Visual Analytics for Comprehending Malware Behavior," in IEEE Access, vol. 10, pp. 99909-99930, 2022, doi: 10.1109/ACCESS.2022.3207782.

[9] Almomani, A. Alkhayer and W. El-Shafai, "An Automated Vision-Based Deep Learning Model for Efficient Detection of Android Malware Attacks," in IEEE Access, vol. 10, pp. 2700-2720, 2022, doi: 10.1109/ACCESS.2022.3140341.

[10] Pattee, S. M. Anik and B. K. Lee, "Performance Monitoring Counter Based Intelligent Malware Detection and Design Alternatives," in IEEE Access, vol. 10, pp. 28685-28692, 2022, doi: 10.1109/ACCESS.2022.3157812.

[11] H. Manthena, J. C. Kimmel, M. Abdelsalam and M. Gupta, "Analyzing and Explaining Black-Box Models for Online Malware Detection," in IEEE Access, vol. 11, pp. 25237-25252, 2023, doi: 10.1109/ACCESS.2023.3255176.

[12] B. Jin, J. Choi, J. B. Hong and H. Kim, "On the Effectiveness of Perturbations in Generating Evasive Malware Variants," in IEEE Access, vol. 11, pp. 31062-31074, 2023, doi: 10.1109/ACCESS.2023.3262265.

[13] T. He, C. Han, R. Isawa, T. Takahashi, S. Kijima and J. Takeuchi, "Scalable and Fast Algorithm for Constructing Phylogenetic Trees with Application to IoT Malware Clustering," in IEEE Access, vol. 11, pp. 8240-8253, 2023, doi: 10.1109/ACCESS.2023.3238711.

[14] M. M. Belal and D. M. Sundaram, "Global-Local Attention-Based Butterfly Vision Transformer for Visualization-Based Malware Classification," in IEEE Access, vol. 11, pp. 69337-69355, 2023, doi: 10.1109/ACCESS.2023.3293530.

[15] Gulatas, H. H. Kilinc, A. H. Zaim and M. A. Aydin, "Malware Threat on Edge/Fog Computing Environments from Internet of Things Devices Perspective," in IEEE Access, vol. 11, pp. 33584-33606, 2023, doi: 10.1109/ACCESS.2023.3262614.