Analyzing the Techniques & Algorithms for Malware Detection Using Deep Learning Neural Networks

Arun Singh Chouhan[1*], Bollamalli Pravalika[2]

[1]Department of CSE, Associate Professor, Malla Reddy University, Hyderbad, India
[2]M.Tech Scholar, Department of CSE, Malla Reddy University, Hyderabad, India
drarunsinghchouhan@gmail.com[1*] , pravalika.bollampalli15@gmail.com[2]

Abstract:

Cybersecurity faces an ongoing and quickly changing challenge from malicious software (malware). Usually, traditional methods of detection rely on static or dynamic analysis methods, neither of which provide complete protection against all versions of malware. With static analysis, malware characteristics like API calls, permissions, and opcodes are extracted without actually executing the malware file; however, these types of analyses are easily obfuscated and packed. While static malware detection methods rely on analyzing variables within a sample without executing it, dynamic malware detection methods utilize the execution of a sample to monitor what happens at runtime (e.g., when a malware sample makes system calls and connects to remote servers). Because of the greater resiliency of dynamic analysis methods over static analyses, they require a significantly higher amount of resources and therefore have created techniques for creating "sandbox" environments where malware can execute. In addition, as with all detection methods, attackers can create unique ways to defeat detection via evasive behaviour and anti-reverse engineering methods. To solve these problems, we created a hybrid malware detection framework that utilizes both types of analysis in combination with deep-learning approaches. This framework utilizes CNNs, LSTMs, and DNNs to capture spatial relationships, temporal relationships, and high- level representations of both types of features within the analysed Windows API dataset(s). Using Evaluation metrics we show that the hybrid detection framework is capable of reliably classifying malware samples as either benevolent or malicious. By providing a hybrid system for the scalable, rapid detection of modern malware threats in today's increasingly complex computer environments, our framework serves as an opportunity to improve malware detection systems.

Keywords: Malware, Hybrid Analysis, CNN, LSTM, DNN.

1. Introduction

In today's world, Windows is the most widely used operating system, and it has become very popular for both personal and enterprise use. Because Windows is so widely used, more attackers are creating new forms of malicious code (e.g. hackers) to attack Windows systems. With the increased number of attacks targeting Windows, attackers have become more sophisticated in how they target Windows systems. Shijo et al.(2015)[1] solution to this problem is to combine both Static Analysis and Dynamic Analysis to improve the effectiveness of detecting known and unknown malware. By combining structural information and actual behaviour, Shijo et al.'s solution produces a more robust and reliable way of detecting malware. By utilizing both static and dynamic features, deep learning techniques have established themselves to be the primary means of identifying malware through effective modelling. Convolutional Neural Networks (CNNs) are used to recognize spatial characteristics from Portable Executables (PE) byte files or Opcode image files, while Recurrent Neural Networks (RNN) or Long Short- Term Memory Networks (LSTM) are used by researchers to model temporal patterns created by API calls.
Because traditional methods, which rely on identifying malware signatures, are no longer effective against obfuscated and polymorphic malware, Anusha Damodaran et al. (2017) [2] argue that these techniques are not sufficient for detecting new types of malware. Therefore, more researchers are turning to automated, intelligent, and trained using deep-learning techniques for effective malware identification. Sequence-based

D. R. Reddy et al. (eds.), *Proceedings of the First International Conference on Advances in Forensics and Cyber Technologies (ICFACT 2025)*, Advances in Computer Science Research 127,
https://doi.org/10.2991/978-94-6239-610-4_42

malware analysis has become popular lately due to the fact that API call sequences represent a true runtime aspect of how an application behaves; therefore, there are many research articles, such as Maniriho et-al. (2023) [7], showing that pattern recognition can provide good discrimination between benign and malicious behaviours by analysing API call sequences. Another recent trend within this general area of research includes the use of PE Metadata; in other words, using static analysis of the header fields, opcodes and imported APIs of an executable to learn about the characteristics of the executable without running it. Malware detection via Recurrent Models utilizing Adversarial Learning was studied for high accuracy and low false positive rates by Owoh et al. (2024) [10] based on API Call Sequences and Recurrent Models. Recently, Hybrid Deep Learning has been introduced by many researchers to combine both Static and Dynamic features to improve performance over either Static or Dynamic alone. According to Gautam et al. (2024) [11], dynamic behaviour detection is imperative for properly characterising polymorphic and metamorphic malware, as these characteristics manifest themselves through dynamic execution traces, in addition to recognising static and evolving patterns associated with malware via static and temporal metrics respectively. Researchers developing machine learning based malware detection (static) models can benefit greatly from this publicly available VirusTotal-scanned dataset of Windows PE samples[20,21].
Muhammad Irfan Yousuf et al. (2022) [22], contain 18551 samples representing 5 different families of malware, which include DLLs, imported functions, and PE header / section characteristics.

2. Related Work

The number of malware attacks against Windows Operating Systems has increased over the years and many of the malware attacks have become much more complicated. Therefore, researchers who study this area have changed their approach when it comes to identifying malware. The older methods of identifying malware (signature based and heuristic scanning) are not as effective for identifying polymorphic, metamorphic, obfuscated and fileless malware. Consequently, the progression of researchers toward using machine learning based and deep learning based techniques to identify malware will result in improved classification accuracy, precision, recall and F1 scores when classifying malware.

The malware detection technology is continuing to develop to identify malware that might be concealed from detection using obfuscation or some form of polymorphic transformation. According to the authors Vyas et al. (2017) [3], they demonstrated that 28 lightweight static PE characteristics could provide an accuracy rating of 98.7%. This allows the fast and low-overhead identification of malware. The absence of a standardized dataset impedes the ability to perform dynamic malware analysis. Therefore, Catak et al. (2019) [4], produced a vast amount of data from the open-source automated malware analysis environment known as Cuckoo Sandbox. The data produced by Cuckoo includes three elements: 1) An API call log; 2) A VirusTotal family label associated with each sample; and 3) A Windows PE File Set which included 7107 samples. Xiang Huang (2021) [5] This paper presents a hybrid visualization method that merges static features with Cuckoo Sandbox–based dynamic behavior, converting both into images for neural-network training. A study of Android malware detection conducted by Ngo et al. (2022)
[6] concluded that deep learning is far superior to traditional ML approaches regarding accuracy, precision, and recall. Maniriho et al. (2023) [7] studied how to analyse sequences of Application Programming Interface (API) calls using a deep learning-based approach. The accuracy and sensitivity rate of their algorithm were very high when classifying malware based on API behaviour, thus demonstrating the importance of API behaviour when classifying malware. This chapter reviews ML techniques for malware detection on the Windows platform, while also explaining how to develop a machine learning (ML) pipeline, some of the problems associated with creating ML training datasets, and up-to-date information about the most sophisticated methods for detecting malware. The chapter also discusses challenges that have arisen since the advent of malware detection using ML algorithms, including concept drift, "adversarial" attacks on ML algorithms, and increased interest in defending against such attacks. According to Owoh et al., 98.9% estimated accuracy was produced using a hybrid model built with a gated recurrent unit (GRU) and has a very low classification error rate due to adversarial training, which improved the model's robustness as a major contributor to this increase in accuracy. The CNN-LSTM hybrid model designed by Gautam et al. (2024) [11] using numerical sequences of API calls was able to achieve 96% accuracy. This was due to dynamic behaviour analysis, which is useful in the identification of both zero-day malware and highly evasive

malware. Mihalache et al. (2024)
[12] This approach aims to overcome the limitations of traditional signature-based detection and enhance the ability to detect novel and evolving threats, including zero-day attacks. Thakur et al. (2024) propose a new hybrid architecture, a combination of CNN and LSTM, for the purpose of detecting malware based on the malware binary as an image, as well as on the features extracted from the PCA. The strength ofthe method lies partly in applying ensemble methods for improved accuracy and efficiency of the detection of malware families.
Bensaoud and Kalita (2024) [14] introduced a CNN-LSTM hybrid model that combines opcode and API-sequence features to improve malware classification accuracy while significantly reducing false positives. Gond and Mohapatra (2025) [15] proposed an Adaptive Learning Framework to handle malware behaviour concept drift, achieving higher long-term accuracy while maintaining stable precision unless major behavioural shifts occur. Zhang (2025) [16] introduced a multi-granularity behaviour- modelling technique that enhances recognition of both fine- and coarse-grained malware behaviours, leading to improved precision and F1-scores in classification. Syed Noman Ali Sherazi (2025) [17] This study introduces a hybrid static–dynamic analysis model for detecting fileless malware, uncovering advanced evasion techniques such as process injection, obfuscated macros, and covert network activity. Shahnawaz et al. (2025) [18], Dynamic malware classification of Windows PE files uses API-call-based images and deep learning to detect malware by analyzing runtime behavioral characteristics, overcoming signature- based limitations. Christofer Fellicious et al. (2025) [19], This study presents a lightweight, order-invariant API-call analysis approach for malware detection, using a large public dataset and machine learning to achieve over 85% F1- score while efficiently identifying malware families.

3. Materials and Methods

This study presents a hybrid deep learning model that integrates static PE metadata and dynamic runtime behavior for malware detection. CNNs learn structural patterns from static features, LSTMs capture sequential API call dependencies, and a DNN fuses both feature types, providing a robust and accurate classification framework for identifying malicious executables.

3.1 Data Set:

The multiple CSV files generated during the course of this research were created from the results of performing both Static and Dynamic analyses of Portable Executable (PE) file formats. The Static Analysis consisted of producing three
(3) independent CSV files: 1) DLLs_Imported.csv which contains a list of the DLL Imports and associated API functions used by each PE Executable; 2) PE_Header.csv which contains header information about structural (image base, entry point, subsystem); and 3) PE_Section.csv containing section-level characteristics (entropy, raw size, virtual size, section names).

The file API_Call_Functions.csv provides characteristics of behavior that occur at run time. API call names, how often they are called, their order, what was input, what returned, and when they were made are examples of what is contained in this document. It documents how the program behaves, the identification of abnormal activity such as file modification, registry modifications and/or network connectivity, and the proper mode to detect malware dynamically and accurately.
The dynamic analysis data was stored in a distinct CSV file that included the sequences of API calls made during runtime by each executable being analyzed and thereby reflects the actual actions the program performed while executing (e.g. creating files, modifying the registry, establishing network connections, etc.) These dynamic features were extracted during the execution of the program and are in addition to the static features that were obtained from the PE file.

The static CSVs have been uploaded to the Python environment via the Pandas library and merged together based on the common columns (sha256, md5, or Name) to form one high-dimensional vector for all executables. The dynamic CSV, which provides information about virtual memory (VM) execution of an executable, was merged with the static data based on the same identifier when possible, otherwise through concatenating by index. Each merged executable record includes complete PE metadata, all API functions

and call sequences observed at runtime. The "Label" column of the merged data set shows the actual class membership of each executable record, whether it is malicious or not.

3.2 Dataset Distribution

This study will employ a dataset with three static feature sets comprised of the same attributes from PE files (the four static features), i.e., DLL Imports, API Call Functions, PE Header Metadata, and PE Section Characteristics. Each record in the dataset has a label column representing the sample type (Malware or Benign) that corresponds to that sample. Each record in the dataset is numerically represented by the label columns. Also included are Dynamic Features in the dataset to capture the executables' behavior while they are running (runtime). These Dynamic Features include the API Sequence calls made during the executable's run.
Samples imported using specific DLL files are identified in the DLLs_Imported.csv using binary indicators (0 or 1) and the SHA-256 hash. A total of n1 samples were identified along with over 300 features, both individually and together, related to those DLLs. The API_Call_Functions.csv provides binary or count based indicators for each executable regarding which APIs were called, providing a supplement for the DLL imports for the static feature set. The PE_Header.csv provides structural features (machine type, number of sections, timestamp, entry point address, and other memory-related features) for each PE header in the sample. The PE_Section.csv provides features associated with the text, data, rdata, rsrc, reloc, and pdata sections identified from the PE header, including virtual and raw sizes, entropy values, and the section characteristics. The type column across all files serves as the combined label for the malware family or benign. By merging the static feature groups across all sample types and the dynamic API call sequences, we built a complete dataset. The sample set has a variety of malware and Benign Software; this gives the ability to train on deep learning methods more effectively across different levels of structure and behaviour characteristics. Class-wise frequency analysis shows a moderate class imbalance; malware families appear with moderate frequency variations, requiring robust performance metrics (macro-F1 and balanced accuracy). Nevertheless, this dataset has sufficient sample sizes of varying kinds in order for Deep Models to be trained effectively. The availability of structural PE features, DLL usage patterns, API call functions, section-level metadata, and dynamic API call sequences contributes to a rich feature space that enhances the discriminative capability of the hybrid CNN–LSTM–DNN model.

3.3 Preprocessing

In this paper, the dataset integrates both static and dynamic features of PE files for malware detection. Static features were extracted from DLL imports, PE headers (image base, entry point, subsystem), and PE sections (entropy, raw size, virtual size, section names), API Call Functions (API call names, call frequency, call sequences, parameters, return values, and timestamps.). Dynamic features were obtained from API call sequences during runtime in a sandbox environment, capturing actual executable behavior. After loading all CSVs into Python using Pandas, missing values were imputed with median values, and identifier or excessively long text columns were removed. Categorical features were processed via one-hot or label encoding, and numeric features were standardized using StandardScaler. Zero-variance features were eliminated, and outliers were filtered with IsolationForest. The cleaned static and dynamic features were reshaped appropriately: static features as input for a CNN, dynamic sequences for an LSTM, and both outputs were fused and passed through a DNN for final classification. A stratified 80-20 split was applied to create training and testing sets.

3.4 Hybrid CNN–LSTM–DNN Architecture

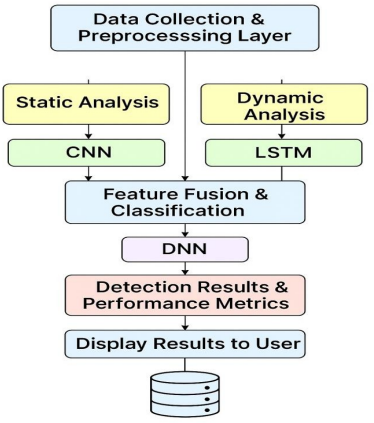
These hybrid deep learning systems for malware detection are a combination of CNNs, LSTMs, and a DNN. The CNN module is applied to static PE file metadata for the capture of local spatial relationships and structural patterns. Two Conv1D layers with 64 and 128 filters extract meaningful features when combined with MaxPooling. Dropout layers reduce overfitting.

In particular, the LSTM component processes API call sequences as sequential dynamic features to model long-range dependencies and structural relationships across execution traces. Although PE metadata are not

inherently temporal, LSTM can learn ordered dependencies of features and spot complex interactions that may be indicative of malicious behavior.

The outputs from CNN and LSTM are fused and passed through a DNN for high-level features' abstraction and classification. Several fully connected layers make up the DNN with ReLU for the activation, hence enabling the model to capture a complex, nonlinear relationship and providing an output. Multi-class outputs are obtained by using softmax, while binary classification makes use of sigmoid activation.

The dataset is preprocessed to handle missing values, normalize features, and separate static and dynamic components. The hybrid model is trained with the Adam optimiser over 30 epochs, with an 80-20 train-test split and early stopping. Performance metrics, including accuracy, precision, recall, F1-score, and confusion



matrix, demonstrate effective malware classification across all classes.

Fig 1: System Architecture

3.5 Model Training

To continuously improve the results obtained during the training period, the authors used a model that had been constructed on top of the original Adam optimiser. This algorithm is well suited to handle multidimensional (and therefore high-dimensional) datasets and is efficient in calculating gradients. The loss function used for this category of neural networks is known as sparse categorical cross entropy. Although there is no specific gain to be achieved by using sparse & categorical cross-entropy loss for a multi-class problem where the labels are integer encoded, the authors believe that the implementation of the Adam optimiser results in more accurate models. Model training consisted of training the model for 30 epochs, using a batch-size of 64 with 20% of the training data being used for validation. Thus, during model training, 20% of the training data served to validate the model, allowing the authors to evaluate the performance of their models and identify potential signs of overfitting. The authors also recorded the training and validation accuracy and lost values to produce learning curves to depict the progress of their model, which demonstrated that the model was learning and improving over time.

3.6 Model Testing

In the evaluated the performance of the model that was created using the training dataset against the test dataset. A variety of metrics were computed to evaluate how well the model performed classification. The primary measures used included overall accuracy, balanced accuracy, the calculated F-scores (both macro and weighted), and a detailed report on the model's classification results. Because many static malware datasets have an unbalanced number of each type of sample, the authors calculated balanced accuracy as

well. To show how well the model could predict the different types of malware, a confusion matrix was generated for the test dataset. The evolution of the training history of the model regarding both accuracy and loss was also plotted over time. All of these evaluations have shown that the trained model exhibited consistency and accuracy, independent of the malware family being identified.

4 Experimental Results

This section provides the results of the tests performed using the Hybrid Classification Framework (HCCF) for malware identification proposed by the researchers, which incorporates CNN, LSTM and DNN architectures. The dataset used to assess this architecture consisted of four distinct types of data on Portable Executable (PE) systems, including: DLL Imports; API calls; PE Header Meta-data; PE Sections. Along with applying static methods to distinguish between types of malware using HCCF, the researchers developed a dynamic sequence of characteristics based on the activity of malware during execution, represented by a series of API calls. This evaluation is aimed at comparing the effectiveness of the HCCF to current Machine Learning (ML) and Deep Learning (DL) Classifiers. Results indicate that the HCCF's Hybrid Architecture has the capability of learning long-term dependencies and structural patterns present in malware and developing sequential relationships based on Dynamic API Call Sequences.

To provide reliable results regarding the effectiveness of the HCCF, the authors employed an 80-20 stratified sample split to create a training and testing dataset and trained the HCCF for 30 epochs.To provide a comprehensive understanding of the performance of the Classifiers, several different Performance Metrics were calculated, including Accuracy, F1 Score, Precision, Recall, Balanced Accuracy, and Confusion Matrix Visualisation. The majority of the malware datasets utilised by the authors have a high level of Class Imbalance.

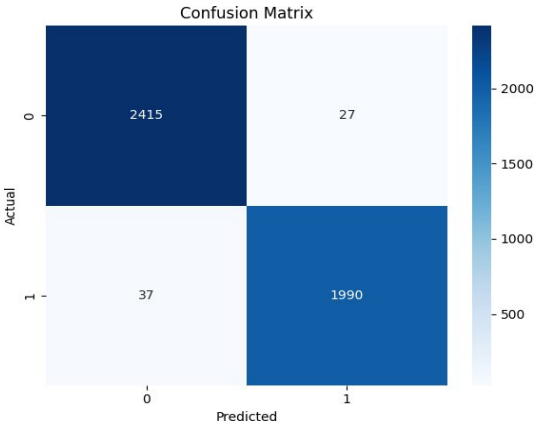
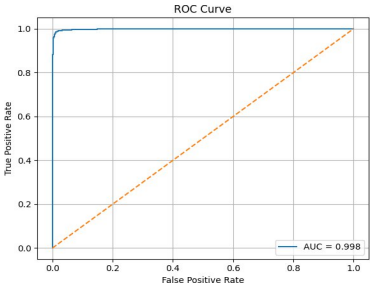
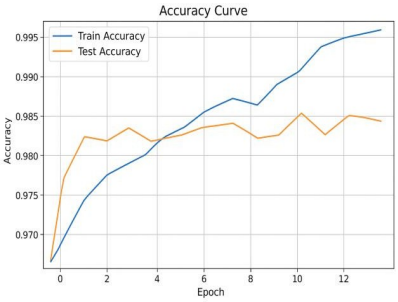


Fig 2: Confusion Matrix

Fig 2 shows A high number of very high diagonal numbers on our confusion matrix indicate that this model was able to effectively separate the benign and malware samples. A total of 2580 benign samples and 2075 malware samples were classified correctly with the model, with very few samples misclassified. The low rate





of error backs up the practicality of using the hybrid CNN- LSTM–DNN model for malware testing.

Fig 3: ROC Curve & AUC

Fig 3 Shows The ROC curve for this model punctuates the extreme power of discrimination that the historical analysis indicates via the enormous area under the curve of 0.9984, indicating a near-perfect ability to discriminate malware versus benign files. This extraordinary vertical slope supports very high true-positive occurrences while at the same time maintains an extremely low false-positive occurrence ratio. Accordingly, this model has a strong predictive capability for detecting malware.

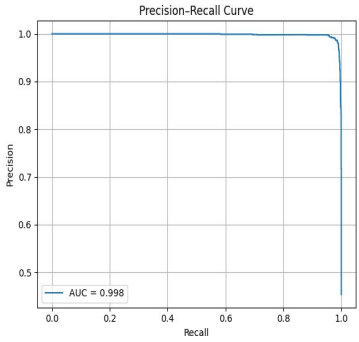


Fig 4: Precision–Recall Curve

Fig 4 indicates Curve produces an impressive PR- AUC score of 0.9981 demonstrating strong effectiveness with regard to class imbalance while achieving high precision and high recall and misclassifications of benign samples as malware are rare occurrences. Also, curve has an upper- right shape indicating the consistent robustness of the method for ambiguous or borderline cases.

Fig 5: Model Accuracy vs. Epochs

Fig 5 Shows The model's ability to learn was seen by the steadily increasing training and testing accuracy curves through all of the epochs. Starting from around epoch 10-12, the training and testing accuracy curves ultimately converge with one another to about 98-99% accuracy. At this point, the training and testing curves indicate that the model has found the balance between being able to learn and being able to generalize. The small space between these two curves provides more evidence that the model did not over-fit, which indicates that the different techniques used for preventing overfitting were effective.

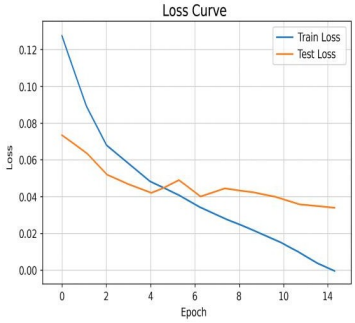


Fig 6: Model Loss vs. Epochs

Fig 6 Shows The decline of both training and testing loss is linear and moving parallel to each other, showing that the learning process is effective and stable. There were slight fluctuations around epoch 10, which would be expected given the stochastic nature of the backpropagation function. The convergence point being achieved is equal to approximately 0.04 which equates to a low percentage (less than 5%) of total prediction error across all epochs considered. Both of these trends are confirmation of strong generalization and support the conclusion that the hybrid architecture will successfully capture the distinguishing characteristics of various types of malware.

Although there are some minor variations in how the performance results were presented for the various models, it was clear that the hybrid model proposed outperformed all of the classifiers in all four of the evaluation metrics-(accuracy, precision, recall, and F1-score)-with an overall accuracy of 98.56%, significantly exceeding that of traditional classifiers such as KNN or XGBoost. The reason the hybrid model outperformed the other models was due to its ability to use the CNN portion to learn local spatial relationships (local pattern) between the PE static features, as well as to use the LSTM portion of the hybrid architecture to learn relationships between PE static features over a long period (long-range sequentially). This unique ability

of the hybrid model allowed for the more precise classification of the various PE static features by using the DNN portion of the model to define more precise decision boundaries. The results of this study clearly illustrate the advantages of using deep hybrid architectures over traditional classifiers based solely on static-analysis methods and demonstrate that the combination of both static and dynamic features will enable more accurate and effective malware detection.

5. Conclusion:

This research illustrates how the sequence of Windows API calls can be used as an effective representation of malware behavior when trying to identify current versions of malware that may evade detection using other methods (e.g., signature-based). The combination of CNN, LSTM and DNN layers in a hybrid deep-learning architecture was able to determine where specific parts of the software are located (spatial pattern), how those parts interact over time (temporal relationship), and also at a higher level how the combination of different types of components results in creating unique characteristics associated with that specific version of malware. This shows that employing a hybrid neural network, in conjunction with Windows API behavioral sequences for Windows security, provides better detection capabilities of automated malware incidents than each of those components individually could do alone.

6. Future scope:

While the current Windows Malware Detection System provides a solid foundation from which to build upon, other improvements could help create a more complete malware detection solution for all platforms (Windows, Linux, Android and Mac OS). Some of these improvements may include; adding support for additional platforms, developing and deploying automated threat detection (real-time) and automated response mechanisms, developing a system that uses a variety of advanced machine learning architectures such as Transformers and GNN to be able to quickly adapt to the changes in the behaviour of malware, increasing the scalability of the system by allowing it to have a federated learning and cloud-based deployment model, providing the ability to have an optimal level of use of the GPU while at the same time developing a lightweight neural architecture to allow for near real-time performance. All of these improvement areas will lead to a more robust and versatile Windows.

References:

1. Shijoa,⁎ , A. Salimb, Integrated static and dynamic analysis for malware detection 1877- 0509 © 2015 The Authors. doi: 10.1016/j.procs.2015.02.149
2. Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H. Austin, Mark Stamp,A comparison of static, dynamic, and hybrid analysis for malware detection (February 2017), Journal of Computer Virology and Hacking Techniques 13(1), DOI:10.1007/s11416-015-0261-z
3. Vyas, R., Luo, X., McFarland, N., & Justice,
C. (2017). Investigation of malicious portable executable file detection on the network using supervised learning techniques. In 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 941–946). https://doi.org/10.23919/INM.2017.7987416
4. Ferhat Ozgur Catak, A Benchmark Api Call Dataset For Windows Pe Malware Classification A Preprint, arXiv:1905.01999v1 [cs.CR] 6 May 2019
5. Xiang Huang, Li Ma, Wenyin Yang, A Method for Windows Malware Detection Based on Deep Learning, March 2021, Journal of Signal Processing Systems 93(1):1-9, DOI:10.1007/s11265-020-01588-1
6. Ngo, M. V., Truong-Huu, T., Rabadi, D., Loo, J. Y., & Teo, S. G. (2022). Fast and efficient efficient malware detection with joint static and dynamic features through transfer learning. arXiv preprint arXiv:2211.13860.
7. Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2023). API-MalDetect: Automated malware detection framework for Windows based on API calls and deep learning techniques. Journal of Network and Computer Applications, 225, 103555. Elsevier. https://doi.org/10.1016/j.jnca.2023.103555
8. Daniel Gibert, Machine Learning for Windows Malware Detection and Classification: Methods, Challenges, and Ongoing Research, July 2024, DOI:10.1007/978-3-031-66245-4_6, In book: Malware (pp.143-173)

9. Andrea Ponte, Dmitrijs Trizna, Luca Demetrio, Battista Biggio, Ivan Tesfai Ogbu, Fabio Roli (2024), SLIFER: Investigating Performance and Robustness of Malware Detection Pipelines https://doi.org/10.1016/j.cose.2024.104264.
10. Owoh, N., Adejoh, J., Hosseinzadeh, S., Ashawa, M., Osamor, J., & Qureshi, A. (2024). Malware detection based on API call sequence analysis: A Gated Recurrent Unit–Generative Adversarial Network model approach. Future Internet, 16(10), 369. MDPI, **https://doi.org/10.3390/fi16100369**.
11. Gautam Karat, Jinesh M. Kannimoola, Namrata Nair, Anu Vazhayil, Sujadevi V G, Prabaharan Poornachandran, CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection, January 2024, Procedia Computer Science 233:492-503, DOI:10.1016/j.procs.2024.03.239, License: CC BY-NC-ND 4.0
12. Mihalache, R., Gavrilu ̧t, D. and Anton, D. Real-Time Deep Learning-Based Malware Detection Using Static and Dynamic Features. DOI: 10.5220/0012316800003636 Paper
13. Preeti Thakur, Vineet Kansal, Vinay Rishiwal (2024) Hybrid Deep Learning Approach Based on LSTM and CNN for Malware Detection. June 2024, Wireless Personal Communications 136(3), DOI:10.1007/s11277-024-11366-y
14. Ahmed Bensaoud, Jugal Kalita, Mahmoud Bensaoud (2024), A survey of malware detection using deep learning, https://doi.org/10.1016/j.mlwa.2024.100546
15. Bishwajit Prasad Gond & Durga Prasad Mohapatra (2025). Deep learning-driven malware classification with API call sequence analysis and concept drift handling. February2025,DOI:10.48550/arXiv.2502.08679, License, CC BY-NC-ND 4.0
16. Zhang, L., & Chen, Y. (2025). A malware- detection method using deep learning to fully extract sequence features. January 2025, 14(1):167, DOI:10.3390/electronics14010167, License: CC BY 4.0
17. Syed Noman Ali Sherazi & Amna Qureshi(2025). Hybrid analysis model for detecting fileless malware. Applied Sciences, 15(1), 314.MDPI. https://doi.org/10.3390/app15010314.
18. Shahnawaz, M., Gond, B. P., & Mohapatra,
D. P. (2025). Dynamic malware classification of Windows PE files using API-call-based images. arXiv preprint arXiv:2505.24231.
19. Christofer Fellicious , Manuel Bischof (2025). Malware detection based on API calls: Order-invariant and lightweight analysis model. arXiv preprint arXiv:2504.11893.
20. Muhammad Irfan Yousuf, A Multi-feature Dataset for Windows PE Malware Classification, Published: 29 December 2022|Version 1|DOI:10.17632/vnj7sxkt53.1
21. Angelo Oliveira, Malware Analysis Datasets: Api Call Sequences by and MalbehavD-V1: A Dataset of API Calls Extracted from Malware and Benign Executable Files in Windows, https://www.kaggle.com/datasets/gautamkarat/ap i-call-sequences, https://doi.org/10.34740/kaggle/ds/3306661