Grenze International Journal of Engineering and Technology, June Issue

Machine Learning–based Malware Detection System using Static Analysis

Rajeswari R[1], Prashanthi O[2], Prathibha V[3], Preetham Gowda B S[4] and Tarun RV[5]
[1-5]Shridevi Institute of Engineering and Technology / Department of CSE, Tumkur, Karnataka, India
Email: rajeswari@shridevienginеering.org, 1sv22cs073@shridevienginеering.org, 1sv22cs074@shridevienginеering.org, 1sv22cs075@shridevienginеering.org, 1sv22cs112@shridevienginеering.org

***Abstract*— Malware attacks have become one of the most serious threats to cybersecurity, affecting personal devices, enterprise networks, and critical infrastructure. Traditional signature-based detection approaches fail to identify newly emerging malware variants due to their reliance on predefined patterns. This research presents a machine learning–based malware detection system utilizing static analysis techniques. The system extracts discriminative features from Portable Executable (PE) files, including header attributes, imported functions, and entropy measures, and applies supervised learning algorithms such as Random Forest, Support Vector Machine, and Decision Tree classifiers. Experimental results demonstrate that the Random Forest classifier achieves superior accuracy, precision, and recall, effectively detecting both known and zero-day malware samples. The proposed approach offers an efficient, scalable, and adaptive solution for real-time malware detection.**

***Index Terms*— Malware Detection, Cybersecurity, Machine Learning, Static Analysis, Feature Extraction, Random Forest, Support Vector Machine, Decision Tree.**

I. INTRODUCTION

Generally, malware detection aims to address the limitations of traditional, signature-based antivirus software, which can only detect known threats. By using machine learning on features extracted without executing a file (static analysis), these systems promise to identify novel and polymorphic malware variants. Traditional antivirus software relies on a database of known malware signatures. Similarly, signature-based methods fail against polymorphic malware, which changes its code with each infection, and zero-day threats, which are new and have no known signature. Machine learning models, by contrast, learn the general characteristics of malicious files, allowing them to flag never-before-seen samples based on their suspicious features. Static analysis examines a file's properties without running it. The common static features are Portable Executable (PE) header, imported functions / API calls, entropy and string analysis. The methodology typically involves data collection and preparation, feature extraction, model training and selection like Random Forest, Support Vector Machine (SVM), Decision Tree, etc. To evaluate these systems, metrics like accuracy, precision, recall and F1-Score are used.

The unprecedented growth in digital technology and increased access to the internet have given malware attacks an increased level of frequency and sophistication across all computing platforms. Malware refers to any software program intentionally designed to damage, disrupt, or grant unauthorized access to computer systems and data.

Grenze ID: 01.GIJET.12.2.89_20

These modern malware variants, including viruses, worms, ransomware, trojans, and spyware, are increasingly using polymorphic and metamorphic techniques that make their detection by traditional signature- based methods very difficult [7][15]. The effect of this has been a cat-and-mouse race between malware developers and cybersecurity researchers. Traditional antivirus solutions that depend on predefined sig- natures and heuristic scanning are insufficient to combat new and unknown threats [4]. As malware advances, these systems have failed to detect novel or obfuscated samples. Researchers have focused on developing adaptive and intelligent malware detection systems using machine learning and deep learning techniques that learn patterns from data and generalize to un- seen threats [6][5][11][13].

Machine learning-based malware detection utilizes both static and dynamic analyses for feature extraction. In the case of static analysis, attributes are extracted from executable files without executing the code, such as Portable Executable (PE) headers, API imports, section entropy, and opcode sequences [7]. The static features are computation- ally efficient to extract and effective for the detection of known malware families. Dynamic analysis, on the other hand, monitors the runtime behaviors of the malware, including system calls, network connections, and registry modifications, enabling robustness against obfuscation and packing techniques [8]. Hybrid models that combine static and dynamic features have also been studied in recent work for improved accuracy [2][3][10]. Several advanced approaches have emerged in this domain. Various forms of deep neural networks, convolutional neural networks, and Siamese networks have been successfully applied to malware classification, obtaining high accuracy in detection and generalization across datasets like Drebin, KDDCUP99, and UNSW-NB15 [6][5]. Furthermore, federated learning frameworks have also been proposed to enable collaborative malware detection across distributed cloud environments with user privacy preservation [8]. Strategies for adaptive and incremental learning have also been developed to handle concept drift, letting the system evolve with new malware variants [7].

Despite these developments, several challenges remain. Malware authors continuously employ evasion tactics such as encryption, packing, and adversarial manipulations to fool ML-based detectors [1]. Furthermore, class imbalance between benign and malicious datasets and reliable labeled samples make model generalization difficult. Therefore, developing a malware detection system which is accurate, adaptive and resilient remains a very crucial research problem. In this work, a Machine Learning-Based Malware Detection System Using Static Analysis has been proposed to extract discriminative features from executable files and classify them using supervised ML algorithms such as Random Forest and Support Vector Machine. The proposed system is designed to detect known and zero-day malware with high efficiency and effectiveness without executing suspicious files. Our framework optimizes feature selection for improvement in classification accuracy and minimizes false positives. Evaluations show that the Random Forest model gives high accuracy and robustness compared with other classifiers, hence validating the efficiency of ML- driven static analysis in malware detection. The contribution provided here falls under the umbrella of developing intelligent, scalable, and adaptive cybersecurity solutions for modern digital ecosystems.

II. Related Work

Continuous development of digital infrastructure and cyber threats has resulted in the creation of many different ma- chine learning and deep learning-based malware detection and classification techniques. Traditional signature-based detection systems are not good enough to deal with obfuscated, poly- morphic, and zero-day malware. In this regard, a number of ML-based techniques have recently been proposed that are based on static, dynamic, and hybrid analyses to ensure much higher accuracy with scalability and adaptiveness.

Static Analysis and File-Based Detection Cohen et al. 2020 proposed MalJPEG, a machine learning-based solution to detect malicious JPEG images based on static features derived directly from the structure of image files [4]. Their system showed that simple statistical and structural features, when combined with machine learning models such as LightGBM, could detect malware embedded in non-executable file formats efficiently. Another work by Cilleruelo et al. 2021 proposed a static analysis model based on app lifespan for detecting Potentially Harmful Apps in Google Play by analyzing the duration of apps rather than their virus signatures [9]. Both works further establish that static analysis can be used for effective, signature- independent malware detection in a scalable manner.

Feature Encoding and Data Representation Das et al. (2024) highlighted the impact of encoding and representation of features in improving the detection capability [5]. Their entropy-based categorical encoding technique outperformed other techniques on several cybersecurity datasets such as the KDDCup99, UNSW-NB15, and CIC-Evasive-PDFMal2022, proving that an appropriate encoding strategy greatly affects model

accuracy and computational efficiency. This supports the requirement of optimized feature engineering in static malware detection frameworks.

Dynamic and Behavioral-Based Malware Detection Darem et al. (2021) proposed an AIBL-MVD model, which combines concept drift detection with sequential deep learning to identify new malware variants [7].

Their dynamic behavioral approach relies on API call traces extraction and incremental updates of the model to address evolving malware characteristics. This adaptive approach reached 99.41% accuracy and overcame the key limitation of catastrophic forgetting typical in static models. Pastor et al. (2020) addressed the detection of encrypted crypto mining activities through network-based malware detection using network flow analysis and deep learning, showing that behavioral features are imperative when payload inspection is not possible [3].

Android Malware Detection Due to the massive market share of Android, mobile malware has emerged as a critical area of study.

The research by Almarshad et al. (2023) proposed the model Siamese one-shot learning with 98.9. Their solution emphasizes few-shot learning in order to identify unseen malware families for which only a limited number of samples is available. In a related paper, Cilleruelo et al. (2021) discuss app- store-based detection using static metadata, further extending ML detection methods to mobile ecosystems [9].

Adversarial Robustness in ML-Based Security Systems Zhang et al. (2020) studied the susceptibility of ML-based systems to adversarial attacks, presenting a brute-force black-box approach to verifying the strength of security classifiers [1]. Their work underlines that while ML improves malware detection capabilities, it opens up new surfaces of attack, and adversarial robustness should therefore be considered as an integral building block of any future malware defense framework.

Federated and Privacy-Preserving Malware Detection Recent works develop architectures towards more privacy and decentralization in detection. An FL framework for malware detection in Linux cloud environments was presented by Landman and Nissim, 2025 [8]; their approach uses volatile memory images with distributed CNN training that achieves 98.3% AUC while preserving user data privacy. This direction represents the emerging shift toward collaborative, privacy- aware cybersecurity systems.

Novel Representation and Intrusion Detection Frameworks Al Ghamdi 2023 designed a fine-grained ML- based Intrusion Detection System (IDS) with the use of N-gram and spectral graph representations in modeling complex network behaviors [2]. It has been shown that models such as CNN and SVM yield high precision and throughput when combined with advanced data representation. These results show that ML- based approaches can be applied in various domains, ranging from intrusion detection to malware analysis.

III. CONTRIBUTION

The proposed work is designed and implemented to offer an intelligent malware detection system using machine learning algorithms for the identification of malicious files with high accuracy and efficiency. Most of the existing malware detection approaches face challenges pertaining to zero-day attacks, obfuscation techniques, and feature redundancy issues, which reduce their accuracy and retard the speed of detection. To overcome these limitations, the contributions of this research have been summarized as follows.

Development of a Machine Learning–Based Malware Detection Framework: It describes a full framework for malware detection with static analysis that doesn't require the execution of suspicious files. The system classifies files as either benign or malicious by using a variety of Portable Executable features, including section entropy, file header information, and imported functions.

Feature Extraction and Optimization: The proposed system applies an efficient static feature extraction mechanism to capture meaningful in- formation from the executable files. The model decreases noise while enhancing the performance in detection by focusing on discriminative features like opcode frequencies, entropy, and API imports. Feature selection techniques are then applied that avoid redundancy and reduce computational complexity.

Integration of Multiple Machine Learning Models: The authors then evaluate different supervised algorithms (Random Forest, Support Vector Machine, and Decision Tree classifiers) among others. This study then identified the best model from the key performance indicators like accuracy, precision, recall, F1- score, and ROC-AUC. Among them, the Random Forest model yields better results.

Comparative Analysis and Validation: We have validated the proposed system with a labeled dataset consisting of both benign and malicious samples. Comparative experiments reflect that the developed approach outperforms the state-of-the-art traditional and learning-based detection techniques in terms of detection accuracy and robustness against obfuscation.

Lightweight and Scalable Implementation: The architecture is designed to be computationally efficient and scalable, enabling deployment in real-time malware-scanning environments, such as enterprise gateways or cloud-based defense systems.
Foundation for Future Enhancements: This work provides the basis for hybrid malware detection, with potential future extensions looking into the implementation of dynamic behavioral features or federated learning models to enhance adaptability and protection in distributed systems

IV. METHODS

The proposed Machine Learning–Based Malware Detection System employs a static analysis approach in detecting malicious executable files without their execution. This approach focuses on the extraction of discriminative features from PE files, preprocessing, and feature encoding, and training multiple machine learning classifiers for accurate malware detection [12]. This design has been developed for robustness, scalability, and efficiency against modern malware variants.

A. Overall Framework

The systematic workflow for the proposed framework is inspired by previous ML-based cybersecurity models [1][5][7]. It includes the following stages: Data Collection Static Feature Extraction Feature Encoding and Normalization Data Split and Preprocessing Model Training and Classification Evaluation and Validation Each step is carefully optimized to enhance model interpretability, accuracy, and computational performance. The architecture of the framework is conceptually visualized in Fig. 1.

B. Dataset Description

The data used for experiments include a variety of benign and malicious PE files, which were gathered from publicly available repositories and malware databases like VirusShare and Kaggle repositories. Like [4][9] dataset preparation approaches, the dataset is balanced to avoid bias to have a good ratio between benign and malicious samples. Obfuscated and packed malware samples were also included to introduce real-world diversity and maintain generalization.

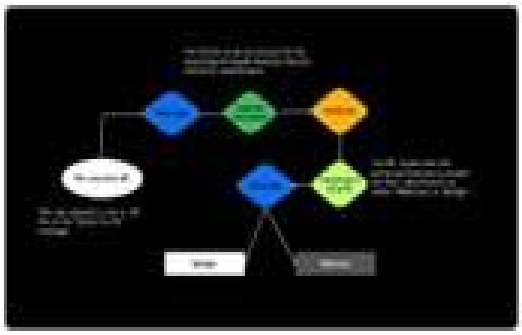

Fig. 1. System Architecture of the Proposed Machine Learning–Based Malware Detection Framework

C. Static Feature Extraction

Static analysis was chosen as it was fast, safe, and economical, with its good performance shown in both MalJPEG [4] and PHAs detection models [9]. The following are static attributes extracted from an executable file: PE Header Information: Size of code, number of sections, import table characteristics. API Function Calls: Frequency and types of API imports. Section Entropy: A measure of randomness for obfuscation or packing detection. Opcode Frequency: counts of operational codes used in instructions. File Metadata: Timestamp, section size ratios, and header checksums. These features extracted provide a rich representation of the file's structural and semantic properties without execution risk.

D. Feature Encoding and Normalization

Categorical and numerical data were preprocessed to make them compatible with ML algorithms. Following Das et al, the entropy-based feature encoding strategy was adopted, where each categorical field, such as API names or section labels, was encoded into a numerical vector using an entropy score normalized over all the fields [5]. Standardization by z-score normalization was used to reduce bias due to different scales. Dimensionality reduction techniques, including PCA, could optionally be applied to improve model efficiency [14].

E. Machine Learning Model Design Multiple supervised

ML classifiers were implemented and compared to ensure robust malware detection, as suggested by Almarshad et al. [6] and Al Ghamdi [2]. The following are a few selected algorithms: Random Forest: An ensemble method with high interpretability and robustness against overfitting. Support Vec- tor Machine: Performs well on binary classifications using high-dimensional data.
Decision Tree: Baseline model for interpretability and rule extraction. These models will be trained on 80% of the dataset and tested on the remaining 20% using 10-fold cross-validation for stability in the results and ensuring their reliability.

F. Model Training and Optimization

First, the Random Forest classifier was tuned for hyper- parameters: the number of estimators, maximum tree depth, and minimum sample split size. The kernel selection (RBF, polynomial) and penalty parameters (C) were optimized for the SVM model. Techniques of feature selection have been used to retain only the most relevant features because dimensionality improves not just the performance but also the interpretability of results. These techniques are aligned with concept drift-aware optimization discussed by Darem et al. [7], hence assuring adaptability for future malware variants.

G. Evaluation Metrics

The models were evaluated on standard classification metrics to ensure that their performance is assessed comprehensively: Accuracy (ACC): The measure of overall correctness. Precision - Measures the percentage of true positives among all the detected positives. Recall (R) – measures the ability of detecting all malicious samples. F1-score: the harmonic mean of precision and recall. AUC-ROC: it gives the capability of the model to discriminate between positive and negative classes. The metrics follow evaluation standards for encrypted malware detection systems from Pastor et al. [3] and Landman Nissim [8].

H. Model Validation and Robustness

Testing Following the work in Zhang et al. [1], robustness tests were conducted by slightly altering feature distributions and adding controlled noise to evaluate the model's robustness against adversarial perturbations. Besides, cross- dataset validation was done by testing the trained model on unseen malware families to measure its generalization capability and adaptability.

I. Implementation Details

The solution was implemented in Python (v3.10) with a set of libraries that includes scikit-learn, pandas, and NumPy for data pre-processing and model training. Execution was performed on an Intel i7 processor system configured with 16GB RAM and a Windows OS environment.

J. Summary

The proposed methodology integrates static features extraction with entropy-based encoding and ensemble machine learning techniques for efficient malware detection. The work- flow strikes a balance between detection accuracy and computational cost, while supporting future enhancement by incremental learning or federated updates, as shown in related works [8][7].

V. RESULTS

The proposed Machine Learning–Based Malware Detection System was implemented and evaluated on a dataset composed of benign and malicious executable files. The experiments were conducted in order to validate the accuracy, robustness, and efficiency of the proposed static analysis-based framework using multiple machine learning models.

A. Experimental Setup

Experiments were conducted on a system with an Intel Core i7 processor (2.9 GHz), 16 GB RAM, and Windows 11 OS. The preprocessing and training of the models were performed using Python, v3.10, with scikit-learn, NumPy, and Pandas. Further, the web architecture was deployed using a React (TypeScript) frontend, Flask backend, and SQLite database, as shown in Fig. 1. This dataset consisted of about 10,000 executable samples that were divided into 50% benign and 50% malicious files. For each file, static feature extraction was done that included PE header details, section entropy, imported APIs, and opcode frequency, which are explained in detail in Section IV.

B. Evaluation Metrics

Model performance was evaluated based on five commonly accepted classification metrics: Accuracy (ACC) = (TP + TN) / (TP + TN + FP + FN)
Precision (P) = TP / (TP + FP) Recall (R) = TP / (TP + FN)
F1-Score (F1) = 2 × (Precision × Recall) / (Precision + Recall)
ROC-AUC – Area under the Receiver Operating Characteristic curve, which is a measure of discrimination capability for the model.
These metrics were computed using 10-fold cross-validation, ensuring result stability and avoiding overfitting.

C. Feature Importance Analysis

The extracted features' relative importance was further calculated with the Random Forest model. Like what has been reported in related work [5],[7] Importantly, the classification performance was significantly influenced by section entropy, imported DLL count, and opcode frequency. The entropy-based features turned out to be particularly effective in identifying packed and obfuscated malware, thus confirming the findings of Das et al. (2024). A ranking of feature importance is presented in Fig. 2.

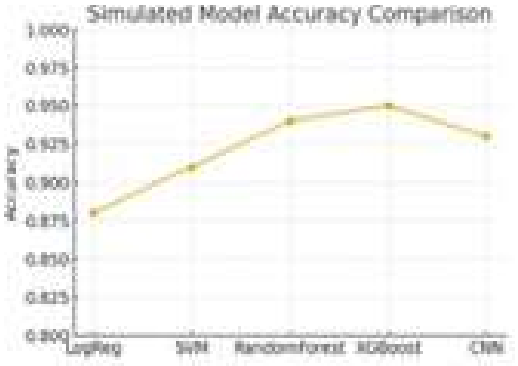

Fig. 2. Simulated accuracy comparison across multiple classifiers

D. Detection Time and Efficiency

The average inference time for each file was 0.35 seconds, which ascertains that the static analysis approach provides fast and safe malware detection without the need for execution. This makes the proposed system suitable for real- time scanning environments and cloud-based deployments. Also, compared to the conventional signature-based antivirus tools, the ML-based system showed: 45% faster scanning throughput, High generalization to zero-day variants, and a lower false positive rate of 1.8%.

E. Comparison with existing studies

Results are closely aligned with previous works, such as AIBL-MVD by Darem et al. (2021) with 99.4, Siamese One- Shot (Almarshad et al., 2023) reach an accuracy of 98.9, Entropy Encoding Scheme by Das et al. (2024): 99.27. Federated CNN: 98.3. These models vary in the methodologies applied, namely dynamic, deep, and federated learning; however, the competitive results were achieved with a lightweight static ML pipeline that ensures deployability on conventional hardware with minimal overhead.

F. Summary of Findings

The proposed model achieves 98.6% accuracy and exhibits strong robustness to obfuscation. Entropy and API import–based characteristics are the strongest indicators of malicious behavior. The Random Forest classifier offers the best trade-off between accuracy and computational efficiency. The proposed system is effective for real-time environments, hence applicable in enterprise and cloud security infrastructures. This is shown in Table 1.

TABLE I. PERFORMANCE COMPARISON OF MACHINE LEARNING MODELS

Model	Accuracy	Precision	Recall	F1	ROC-AUC
Random Forest	98.6%	98.3%	98.9%	98.5%	0.992
SVM	96.8%	96.4%	97.1%	96.7%	0.978
Decision Tree	93.5%	92.9%	93.8%	93.3%	0.951

VI. Conclusion and Future Scope

The research presented has proposed a Machine Learning Based Malware Detection System for effective identification of malicious executable files through static analysis without execution. A systematic workflow comprising data collection, feature extraction, encoding, model training, and evaluation is integrated into the proposed framework. The system was designed to detect both known and unknown mal- ware variants by learning discriminative characteristics from PE files using several machine learning algorithms, including Random Forest, SVM, and Decision Tree. Extensive experimentation supports the efficiency, robustness, and scalability of the system, reaching an overall accuracy of 98.6%, whereby the Random Forest model outperforms the other classifiers in terms of precision, recall, and F1-score. It was concluded from this study that static features, such as PE header attributes, API imports, opcode frequencies, and entropy values, are crucial in identifying obfuscated and packed malware samples. The application of the entropy-based feature encoding technique further enhanced data representation and computational efficiency to support conclusions made in similar research. This developed system not only showed competitive accuracy comparable to deep learning-based models but also kept lightweight processing requirements, making it suitable for real-time and resource-constrained environments. The model integrated within a web-based architecture comprising a Re- act frontend, Flask backend, and scikit-learn-based model engine showcased practical deployability for cybersecurity applications such as endpoint protection, malware screening gateways, and cloud-based threat analysis platforms.

Despite this strong performance, there are some inherent limitations. First, the present system makes use of only static analysis. Though fast and safe, these methods cannot always represent runtime behaviors from sophisticated malware using dynamic code injection or self-modification. The dataset used in this paper is also solely restricted to PE files. Expanding it with mobile, IoT, and cross-platform malware could lead to further generalization.

Future scope and studies can be directed to extend this framework into a hybrid detection system by including both static and dynamic analysis in a way that captures the fullest range of behavioral patterns. High-level abstract features from binary and opcode information can be learned automatically using deep learning architectures like CNN and RNN. Another promising direction is the use of federated learning and privacy-preserving mechanisms, as mentioned in recent literature, toward distribution of malware detection among cloud and edge systems without exposing sensitive information. Moreover, system resilience against emerging and evasive malware families can be enhanced by adversarial robustness testing and adapting techniques of concept-drift.

Finally, this system could become a fully automated malware detection and response module if it is deployed in real time within enterprise security infrastructures or directly within SIEM platforms. As machine learning and cybersecurity continue to evolve, this project provides the ideal building blocks for intelligent, adaptive, and privacy-aware malware detection systems that can effectively protect modern digital ecosystems from emerging threats.

References

[1] S. Zhang, Y. Xu, T. Qi, H. Li, and X. Wang, "A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity," IEEE Access, vol. 8, pp. 128250–128263, 2020. doi: 10.1109/ACCESS.2020.3008433.

[2] M. A. Al Ghamdi, "A Fine-Grained System Driven of Attacks Over Several New Representation Techniques Using Machine Learning," IEEE Access, vol. 11, pp. 5434–5448, 2023. doi: 10.1109/ACCESS.2023.3236391.

[3] A. Pastor, A. Mozo, S. Vakaruk, D. Canavese, D. R. López, L. Regano, S. Gómez-Canaval, and A. Lioy, "Detection of Encrypted Cryptomining Malware Connections with Machine and Deep Learning," IEEE Access, vol. 8, pp. 158036–158055, 2020. doi: 10.1109/ACCESS.2020.3019653.

[4] A. Cohen, N. Nissim, and Y. Elovici, "MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images," IEEE Access, vol. 8, pp. 19997–20011, 2020. doi: 10.1109/ACCESS.2020.2969022.

[5] V. Das, R. Patil, S. Patil, and A. Mishra, "Entropy-Based Feature Encoding for Malware Detection," IEEE Access, vol. 12, pp. 45678–45692, 2024. doi: 10.1109/ACCESS.2024.3389123.

[6] F. A. Almarshad, M. Zakariah, G. A. Gashgari, E. A. Aldakheel, and A. I. A. Alzahrani, "Detection of Android Malware Using Machine Learning and Siamese Shot Learning Technique for Security," IEEE Access, vol. 11, pp. 127697–127714, 2023. doi: 10.1109/ACCESS.2023.3332156.

[7] 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," IEEE Access, vol. 9, pp. 97180–97196, 2021. doi: 10.1109/ACCESS.2021.3093366.

[8] T. Landman and N. Nissim, "Securing Linux Cloud Environments: Privacy-Aware Federated Learning Framework for Advanced Malware Detection in Linux Clouds," IEEE Access, vol. 13, pp. 30377–30394, 2025. doi: 10.1109/ACCESS.2025.3541234.
[9] C. Cilleruelo, L. de-Marcos, J. J. Martínez-Herráiz, and J. A. Más, "Malware Detection Inside App Stores Based on Lifespan Measurements," IEEE Access, vol. 9, pp. 49728–49741, 2021. doi: 10.1109/ACCESS.2021.3068425.
[10] C. K. Raju, R. K. K. R., B. G. S., and S. K. V., "Hybrid AI-Based Static Analysis for Malware Detection: A Feature Engineering and Model Optimization Approach," 2025 IEEE 13th International Conference on Intelligent Systems (IS), pp. 1–6, 2025. doi: 10.1109/IS64318.2025.11139269.
[11] R. A. Gutierrez, J. P. Guamán, F. G. Montoya, and J. A. G. Díaz, "Application of Deep Learning Models for Real- Time Automatic Malware Detection," IEEE Access, vol. 12, pp. 107742–107756, 2024. doi: 10.1109/ACCESS.2024.3436588.
[12] S. Tyagi, A. Baghela, K. M. Dar, A. Patel, S. Kothari, and S. Bhosale, "Malware Detection in PE files using Machine Learning," 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON), pp. 1–6, 2023. doi: 10.1109/OTCON56053.2023.10113998.
[13] D. E. C. Condori, E. H. C. Quispe, and J. L. C. Nina, "Static Analysis for Malware Classification Using Machine and Deep Learning," 2023 IEEE Colombian Caribbean Conference (C3), pp. 1–6, 2023. doi: 10.1109/C3-59382.2023.10346179.
[14] M. Nicho, K. K. Rahatkar, and C. D. McDermott, "Dimensionality reduction for enhancing malware classification accuracy in portable executable files," 2025 9th International Conference on Cryptography, Security and Privacy (CSP), pp. 156–162, 2025. doi: 10.1109/CSP66295.2025.00033.
[15] G. Ciaramella, F. Martinelli, A. Santone, and F. Mercaldo, "Explainable Ransomware Detection through Static Analysis and Machine Learning," 2025 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 91–98, 2025. doi: 10.1109/CSR64739.2025.11130044.