MALWARE DETECTION SYSTEM BASED ON PE FILE AND URL ANALYSIS USING MACHINE LEARNING

[1]Prof B Prajna, [2]Aditi Todi, [2]Adapureddi Saranya, [2]A. Sharon Sanjana, [2]Amujuri Keerthana

[1]Professor, Head of Department (M.Tech, Ph.D.), [2]BTech Student
[1]Department of Computer Science Engineering, Andhra University College of Engineering for Women, Visakhapatnam,
[2]Department of Computer Science Engineering, Andhra University College of Engineering for Women, Visakhapatnam
[2]adititodi5678@gmail.com

Abstract—With the exponential increase in internet-connected systems and services, cyber threats have grown in complexity and scale. One of the most prevalent forms of cyber threats is malware, often delivered through executable files or malicious URLs. This paper presents a comprehensive malware detection system that integrates two detection mechanisms—Portable Executable (PE) file analysis and URL analysis—leveraging machine learning techniques for classification. The PE file detection module extracts structural and statistical features using pefile [8], while the URL scanner relies on lexical analysis with TF-IDF [3] vectorization. A Flask web application serves as the user interface, allowing users to upload executable files or input URLs for malware detection. The system achieves high accuracy using Random Forest and Logistic Regression models, respectively, and demonstrates the practicality of ML-based approaches in proactive threat detection.

Keywords— Malware Detection, Portable Executable, URL Analysis, Machine Learning, Flask, TF-IDF [3], PE Header Features, Logistic Regression, Random Forest.

I. INTRODUCTION

In today's digital era, malicious software—commonly known as malware—poses one of the most critical security challenges. Cybercriminals exploit executables and deceptive URLs to deploy malware across devices, leading to data breaches, system downtime, and financial losses. Conventional signature-based antivirus solutions struggle to cope with new and obfuscated malware variants. Hence, the integration of Machine Learning (ML) techniques has emerged as a promising solution for detecting previously unknown threats based on behavioral and structural features.

This paper introduces a dual-mode malware detection system that performs static analysis of executable files and lexical analysis of URLs using well-established ML algorithms. The approach is cost-effective, interpretable, and capable of adapting to novel threats with retraining.

II. LITERATURE REVIEW

Traditional malware detection relies heavily on signature databases. While efficient for known threats, these approaches fail to detect polymorphic or zero-day attacks. Recent research suggests that machine learning offers enhanced generalization by learning patterns in file headers, entropy, and string distributions [1]. Similarly, lexical and statistical URL features [2], [3] can be used to detect phishing or command-and-control domains [2].

Tools like PEiD, VirusTotal, and Snort exist, but most are either signature-based or network-dependent. Our approach, in contrast, combines offline static PE analysis with URL-based lexical classification in a user-friendly and modular web application.

III. SYSTEM ARCHITECTURE

The system comprises the following components:

1. PE File Analyzer: Parses the Portable Executable (PE) structure using pefile [8], extracting features from headers, sections, imports, and entropy.
2. URL Analyzer: Tokenizes URLs using lexical patterns and vectorizes them using TF-IDF [3].
3. Model Training Module: Trains two ML classifiers: Random Forest for PE files and Logistic Regression for URLs [1], [4].
4. Flask Web Interface: Allows file uploads or URL inputs and shows predictions.

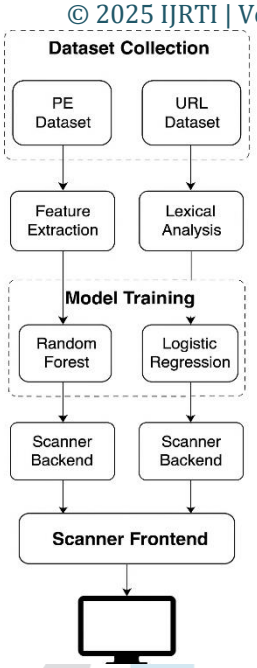


Figure 1: System Architecture Overview

IV. METHODOLOGY

• ***PE Dataset:*** Derived from Kaggle's malware classification dataset.
• ***URL Dataset:*** Sourced from an open-source collection of labeled URLs.

Feature Extraction:

PE File:

• Header fields, Optional headers, Section Entropy, Import/export counts

Example features include:

['DllCharacteristics', 'Machine', 'Characteristics', 'MajorSubsystemVersion', 'Subsystem', 'VersionInformationSize', 'ImageBase', 'ResourcesMaxEntropy', 'ResourcesMinEntropy', 'SectionsMaxEntropy', 'SizeOfOptionalHeader', 'SizeOfStackReserve', 'MajorOperatingSystemVersion']

URL:

• Tokenization via sanitization method
• TF-IDF [3] vectorization Dataset Collection

Example:

TfidfVectorizer(tokenizer=<function sanitization at 0x000002162E1C4D60>)

Model Training:

• PE Model: RandomForestClassifier(n_estimators=50) trained on selected features using ExtraTreesClassifier for feature importance.
• URL Model: LogisticRegression(solver='lbfgs ') trained on TF-IDF [3] matrix.

Models are saved using joblib.

V. IMPLEMENTATION

PE File Detection Workflow:

• Train model using PE.py
• Extract and predict with PE_main.py

URL Detection Workflow:

• Train model using URL.py
• Predict with url_main.py

Web Interface:

• Flask-based with endpoints: '/', '/scan_pe', '/scan_url'

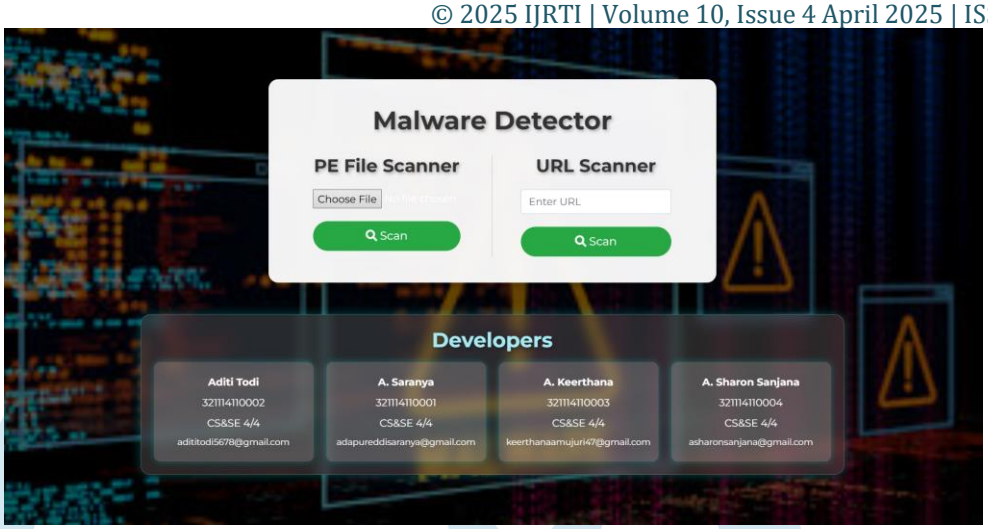


Figure 2: Web Application Homepage (index.html)

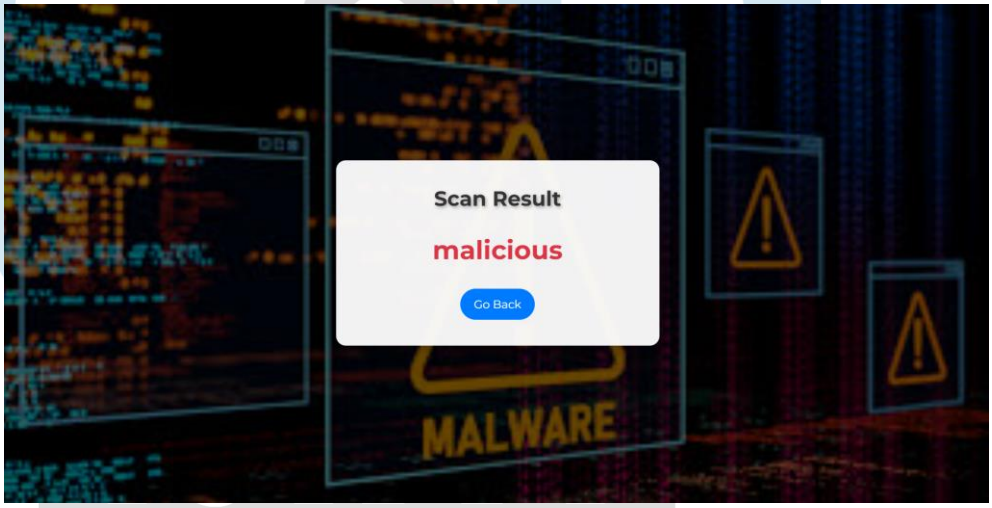


Figure 3: Prediction Output Interface (result.html)

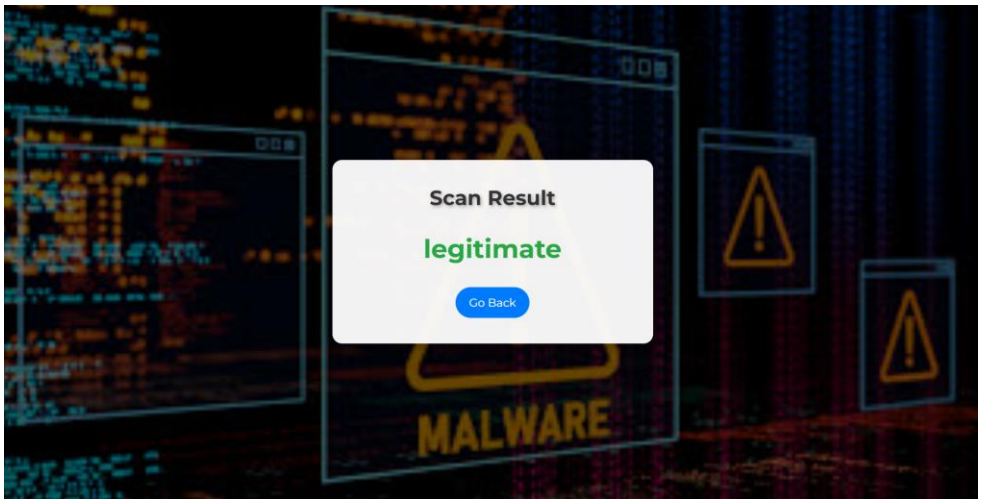


Figure 4: Prediction Output Interface (result.html)

VI. RESULTS AND EVALUATION

Accuracy:

• PE File Model: ~**99.39%**

Table 1: RandomForest classification report

Column	Precision	Recall	F1-score	Support
Malicious	0.9961108	0.995258	0.995684	28050
Legitimate	0.9889241	0.990905	0.989913	11984
Accuracy	0.9939551	0.993955	0.993955	0.993955
Macro avg	0.9925174	0.993082	0.992799	40034
Weighted avg	0.9939594	0.993955	0.993957	40034

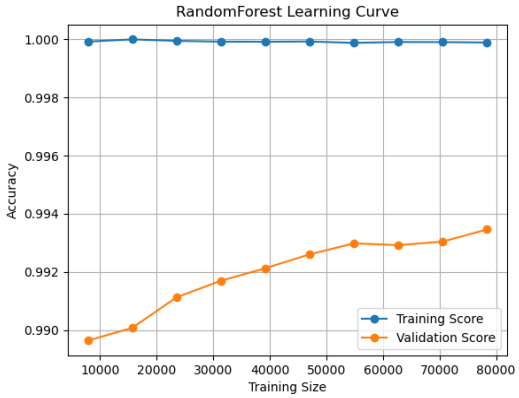


Figure 5: RandomForest Learning Curve

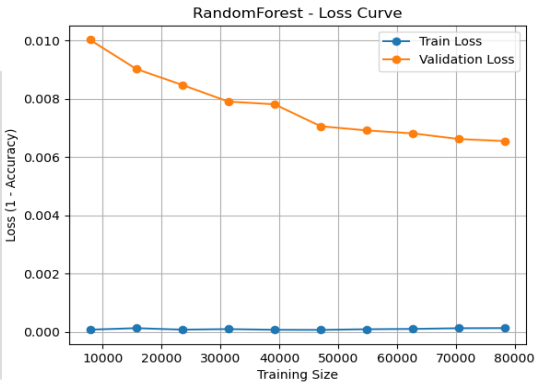


Figure 6: RandomForest Loss Curve

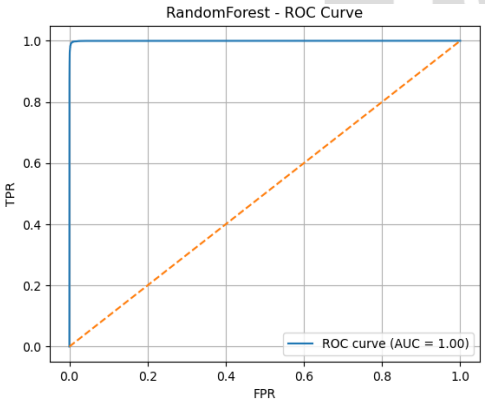


Figure 7: RandomForest ROC Curve Curve

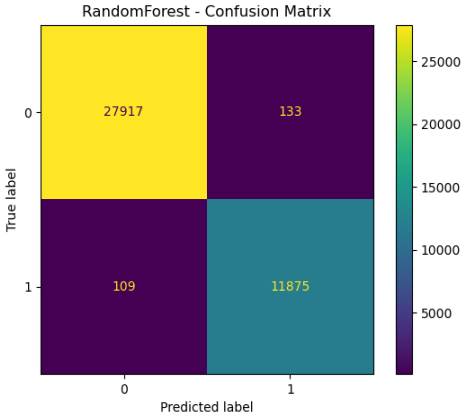


Figure 8: RandomForest Confusion Matrix

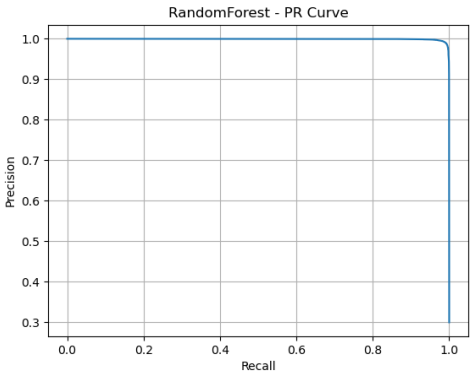
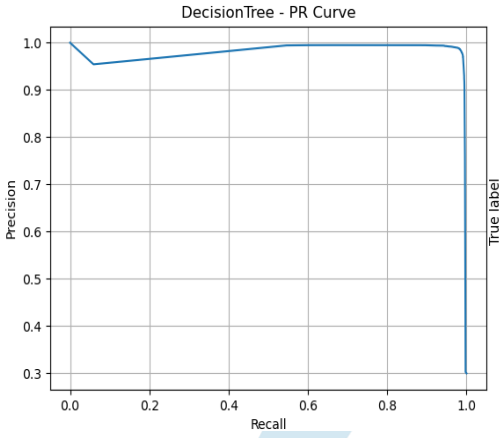
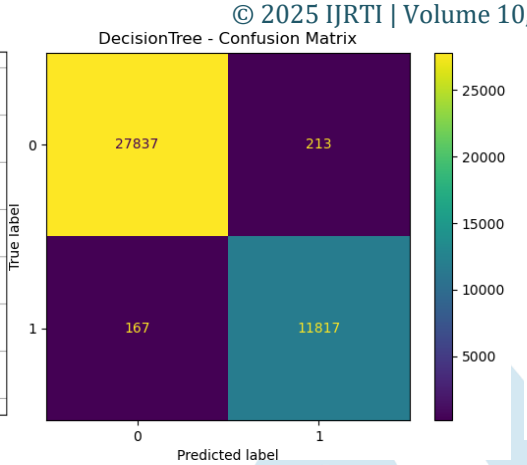


Figure 9: RandomForest PR

Table 2: DecisionTree classification report

Column	Precision	Recall	F1-score	Support
Malicious	0.9940366	0.992406	0.993221	28050
Legitimate	0.9822943	0.986065	0.984176	11984
Accuracy	0.9905081	0.990508	0.990508	0.990508
Macro avg	0.9881654	0.989236	0.988698	40034
Weighted avg	0.9905216	0.990508	0.990513	40034





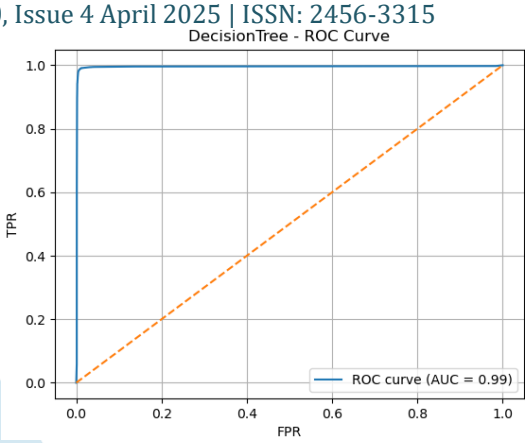


Figure 10: DecisionTree PR Curve

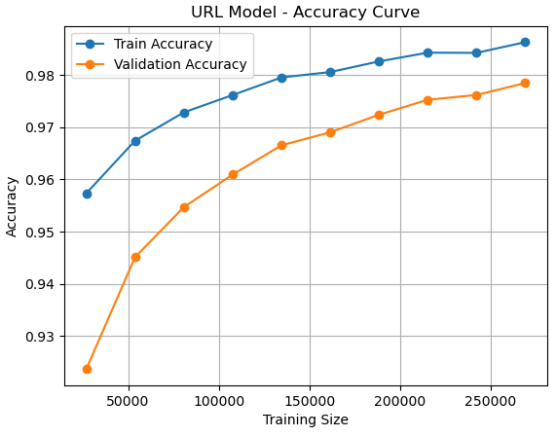
Figure 11: DecisionTree Confusion Matrix

Figure 12: DecisionTree ROC Curve

• URL Model: ~**98.26%**

Table 3: LogisticRegression classification report

Column	Precision	Recall	F1-score	Support
Bad	0.9886869	0.951871	0.96993	28278
Good	0.9760674	0.994482	0.985189	55815
Accuracy	0.9801529	0.980153	0.980153	0.980153
Macro avg	0.9823771	0.973176	0.977559	84093
Weighted avg	0.9803109	0.980153	0.980057	84093



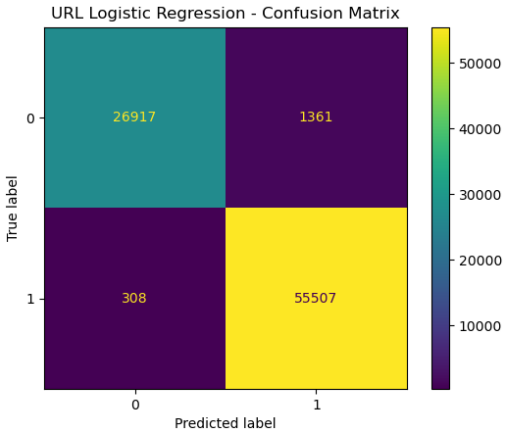
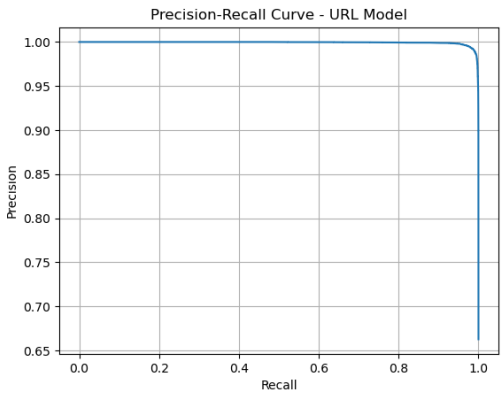
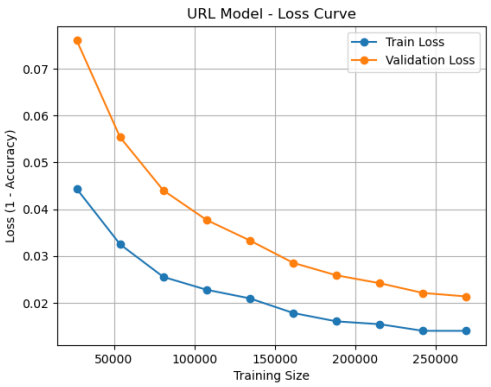


Figure 13: LogisticRegression Accuracy Curve

Figure 14: LogisticRegression Confusion Matrix





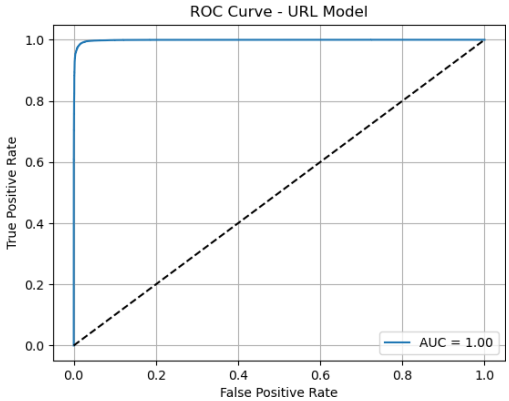


Figure 15: LogisticRegression PR Curve

Figure 16: LogisticRegression Loss Curve

Figure 17: LogisticRegression ROC Curve

VII. CONCLUSION:

This research demonstrates an effective malware detection mechanism integrating static PE file analysis and lexical URL scanning. By combining well-known ML classifiers with precise feature extraction, the system achieves high detection accuracy. Future improvements may include dynamic analysis and integration with cloud threat intelligence

VIII. FUTURE SCOPE

Deep learning models like LSTM
Database integration
API endpoints
Extend detection to other formats like .js, .vbs, and macro-enabled .doc files

IX. ACKNOWLEDGMENT

We would like to extend our sincere gratitude to the Department of Computer Science and Systems Engineering, Andhra University College of Engineering for Women, for their support and guidance throughout this project.

We are especially thankful to our faculty mentors and lab instructors for their constructive feedback, and to our peers for their helpful suggestions during development and testing.

This project has been an insightful learning experience, and we appreciate the open-source communities behind scikit-learn [7], pefile [8], and the publicly available datasets that enabled our research.

REFERENCES

[1] Anderson, H. S., & Roth, P. (2018). EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv:1804.04637. https://arxiv.org/abs/1804.04637
[2] Sahoo, D., Liu, C., & Hoi, S. C. H. (2017). Malicious URL Detection using Machine Learning: A Survey. arXiv:1701.07179. https://arxiv.org/abs/1701.07179
[3] Reyes-Dorta, N., Caballero-Gil, P., & Rosa-Remedios, C. (2024). Detection of Malicious URLs Using Machine Learning. Wireless Networks, 30, 7543–7560 https://doi.org/10.1007/s11276-024-03700-w
[4] Connors, C., & Sarkar, D. (2022). Machine Learning for Detecting Malware in PE Files. arXiv:2212.13988. https://arxiv.org/abs/2212.13988
[5] Patgiri, R., Biswas, A., & Nayak, S. (2021). DeepBF: Malicious URL Detection Using Learned Bloom Filter and Evolutionary Deep Learning. arXiv:2103.12544. https://arxiv.org/abs/2103.12544
[6] Aslam, S., Aslam, H., Manzoor, A., Hui, C., & Rasool, A. (2024). AntiPhishStack: LSTM-based Stacked Generalization Model for Optimized Phishing URL Detection. arXiv:2401.08947. https://arxiv.org/abs/2401.08947
[7] Scikit-learn documentation – https://scikit-learn [7].org
[8] pefile [8] module – https://github.com/erocarrera/pefile [8]