© February 2025 | IJIRT | Volume 11 Issue 9 | ISSN: 2349-6002

Multi-Class Malware Detection using modified GNN and Explainable AI

Shubham Kumar[1], Vishwatej Khot[2], Sagar Bhat[3], Atharva Ghare[4], Ms. Reshma Kapadi[5]
[1,2,3,4,5] Dept. of Computer Engineering, MMCOE, Karve Nagar, Pune, India

Abstract: **As malware becomes increasingly sophisticated, traditional detection techniques, such as signature-based systems, face challenges in effectively identifying novel and polymorphic threats. This research introduces a multi-class malware detection framework utilizing a modified Graph Neural Network (GNN) architecture. By converting bytecode and assembly (ASM) files into image representations, the proposed method enhances feature extraction and improves malware characterization. The method utilizes Graph Neural Networks (GNNs) to capture relationships within the data, allowing for a more flexible and resilient detection system. Additionally, Explainable AI (XAI) techniques, such as Grad-CAM, are integrated to improve transparency by offering visual explanations of the model's decision-making process. The goal is to create a malware detection system that ensures both high accuracy and interpretability, enhancing its reliability and practicality in cybersecurity.**

1. INTRODUCTION

The increasing sophistication and adaptability of modern malware pose significant challenges for cybersecurity experts. Traditional detection methods, such as signature-based systems, are becoming less effective as they primarily focus on identifying known threats. These approaches depend on predefined malware signatures, which can be easily bypassed by contemporary malware employing polymorphic and metamorphic techniques to modify its code structure. Consequently, conventional methods often struggle to detect emerging and evolving malware variants.

In recent years, deep learning has gained considerable attention for malware detection due to its capability to automatically extract complex patterns from large datasets. Among deep learning models, Graph Neural Networks (GNNs) have shown great promise in malware analysis, as they can effectively capture relationships between different components of malware files, such as bytecode and assembly (ASM) instructions. By transforming these files into graph-based representations, GNNs can identify intricate dependencies that traditional techniques may fail to recognize.

Furthermore, the increasing complexity of AI models has raised concerns regarding their "black box" nature, particularly in critical fields like cybersecurity. While deep learning models can deliver high accuracy, their decision-making processes often lack transparency, making it difficult for users to interpret how specific predictions are made. This issue is particularly significant in cybersecurity, where clarity and accountability are essential for trust.

To address these challenges, our research proposes a multi-class malware detection framework that integrates GNNs with Explainable AI (XAI) techniques. Specifically, we introduce DeeperGCN, a modified GNN architecture, to analyze malware encoded as image representations of byte and ASM files. To enhance interpretability, we incorporate Grad-CAM, a visualization tool that highlights the most influential regions of input data contributing to the model's decision. This approach aims to provide cybersecurity professionals with clearer insights into why specific files are classified as malicious, improving the system's transparency and reliability.

This paper outlines the methodology and design of our proposed malware detection framework. Additionally, we discuss the challenges of implementing explainable deep learning models in cybersecurity and explore how our approach can help mitigate these issues, ultimately leading to a more effective and interpretable detection system.

2. LITERATURE SURVEY

The Utilization of Transformers and Self-Supervised Learning (SSL) for Dynamic Malware Detection employs self-supervised language models for detecting malware. Findings suggest that self-supervised models perform comparably to supervised ones, with autoregressive models showing better performance when using minimal labeled data. The key advantage is a reduced dependency on large labeled datasets, but masked language models tend to underperform in strict false-positive conditions. Performance evaluations through ROC curves show that the autoregressive model achieves detection rates close to fully supervised models, with only a 3% drop.

A Multi-Task Deep Learning Model for Malware and Device Classification Using IoT Datasets applies late feature selection across multiple modalities to enhance classification accuracy. This approach improves both malware and device classification, demonstrating strong performance with limited data. Feature selection further boosts accuracy, though the method has high computational complexity due to the size of the feature sets. Cross-validation accuracy reached 74.25% for binary classification, with stable results across different data modalities.

The Similarity Encoding with Deep Learning for Detecting Adversarial Examples in IIoT Malware introduces a method to detect adversarial attacks on Industrial IoT malware. Smaller K values enable faster detection but decrease reliability, leading to higher false-positive rates. This technique allows efficient early detection with minimal computational requirements. Performance assessments through ROC curves highlight the trade-off between K values and detection accuracy.

A Survey on ILP Techniques for Explainable AI explores Probabilistic ILP (PILP), Meta-Interpretive Learning (MIL), and Differentiable ILP (∂ILP). ILP proves effective in generating interpretable explanations while remaining robust in noisy environments. It works efficiently with small datasets and provides symbolic explanations, but it struggles with large datasets and complex relationships. Performance comparisons show that Popper excels in accuracy, while Aleph offers faster processing but lower reliability in complex tasks.

The Iterative Feature Learning for Malware Detection Using Graph-Based Feature Selection constructs new features through arithmetic operations, evaluates them based on probability distribution, and selects high-quality, uncorrelated features. This technique enhances classifier accuracy, particularly when using random forests, and outperforms state-of-the-art methods in 9 out of 15 datasets. It features fast convergence, low sensitivity to input parameters, and explainable feature representation, though computational demands increase with looser feature selection thresholds. Random forests achieved a median accuracy of 81.23% across test cases.

A Balanced Stacked Random Forest Classifier for DoH Attack Detection utilizes the CIRA-CIC-DoHBrw-2020 dataset and employs SHAP values for interpretability. The model effectively classifies malicious DoH attacks with high accuracy (99.91% F1-score, precision, and recall). Its primary advantage is high precision in detecting malicious traffic while providing transparency through SHAP-based feature explanations. However, it is more complex than simpler detection techniques. The model demonstrated 99.9% accuracy in distinguishing DoH traffic from regular HTTPS connections.

NNVisBuilder: A Toolkit for Rapid Prototyping of Neural Network Visualization Interfaces offers a framework for visualizing neural networks, integrating seamlessly with PyTorch. It allows users to create customizable visualizations for debugging and network interpretation. The toolkit facilitates rapid prototyping and reduces coding effort, though performance may degrade in large-scale networks with complex visual designs. The usability of NNVisBuilder was demonstrated through LSTMVis and CNN visualization interfaces, comparing favorably with other analytical tools.

Lastly, the Evaluation of Covert Timing Channel Detection Methods assesses classical approaches such as ε-similarity and compressibility scores against newer techniques like GAS and SnapCatch. The study introduces ε-κlibur to test the limitations of these methods, revealing that all four detection methods can be bypassed through minor modifications in covert channel behavior. The enhanced ε-similarity heuristic successfully detects both classical and ε-κlibur covert channels. However, existing detection techniques are not robust against even simple variations. ROC curve analysis shows that ε-κlibur significantly reduces detection accuracy for GAS and SnapCatch

3. METHODOLOGY

3.1 System Architecture Diagram

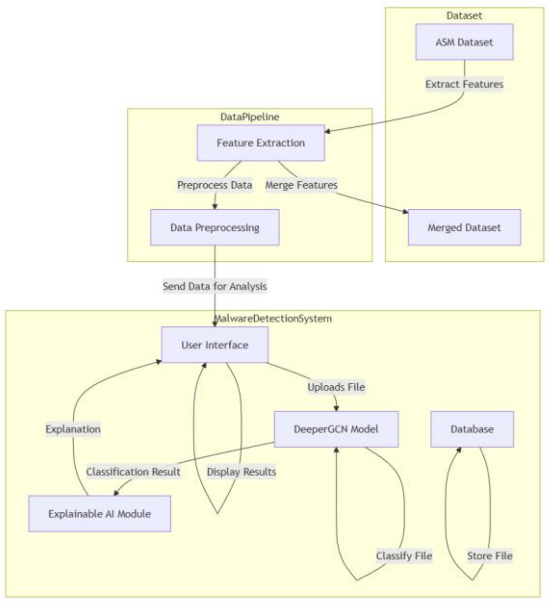

Fig 3.1

The proposed Malware Detection System is designed to automate malware detection and provide human-readable explanations. It comprises three key subsystems:

1. Dataset – Contains various datasets, such as ASM and merged datasets, for feature extraction and analysis.
2. Data Pipeline – Responsible for feature extraction and data preprocessing, ensuring the input is optimized for classification.
3. Malware Detection System – Performs file classification using a machine learning model and generates explanations to enhance interpretability.

This system streamlines malware analysis by integrating preprocessing, classification, and explainability, improving detection accuracy and transparency.

3.2 Dataset

The system utilizes two types of datasets:

The ASM Dataset contains features extracted from the assembly code of executable files, offering a structural view of the binary's operations.

These datasets are combined to form a Merged Dataset, ensuring that both structural and raw data features are incorporated. This merging process enhances the model’s ability to classify files more accurately by leveraging comprehensive feature representation.

3.3 Data Pipeline

The Data Pipeline is responsible for processing the datasets before they are sent for classification. This pipeline consists of two primary stages:

Feature Extraction: In this stage, relevant features are extracted from both the ASM datasets. These features are essential for accurately training the machine learning models.

Data Preprocessing: Following feature extraction, the data is preprocessed through cleaning, normalization, and transformation to ensure it is properly formatted for analysis. Once preprocessed, the data is fed into the malware detection system for further evaluation.

3.4 Malware Detection System

The core of the system is the Malware Detection System, which consists of multiple components working together to detect and explain malware. This system includes the following:

User Interface (UI): The user interface allows users to upload files for malware detection. It is designed to be intuitive, enabling easy interaction with the system.

DeeperGCN Model: The classification process is performed by the Deeper Graph Convolutional Network (DeeperGCN), a deep learning model designed to detect patterns and relationships within the data that indicate whether a file is benign or malicious. The classification result is returned to the UI.

Explainable AI Module: To enhance transparency, the system includes an Explainable AI Module that generates explanations for classification results. This module helps users understand why a file has been classified as malware or benign.

Database: The Database component stores relevant data, including uploaded files, classification results, and generated explanations. This allows for future reference and further analysis.

3.5 Workflow

The system operates as follows:

1. A user uploads a file through the User Interface.
2. The file is sent to the DeeperGCN Model for classification.
3. The DeeperGCN Model returns a classification result (malware or benign) to the UI.
4. If an explanation is requested, the Explainable AI Module generates a detailed explanation, which is displayed to the user.
5. All relevant data, including the uploaded file and classification result, is stored in the Database for future use.

4. APPLICATIONS

The Multi-Class Malware Detection System using modified GNN and Explainable AI has several practical applications in cybersecurity:

1. Enterprise Security: The system can be integrated into corporate networks for real-time malware monitoring and classification, improving threat response.
2. Incident Response: It provides clear explanations for flagged files, aiding forensic investigations and helping analysts understand the origins of attacks.
3. Cloud Security: The system detects malware in cloud environments, ensuring continuous protection and immediate threat identification.
4. Regulatory Compliance: It facilitates adherence to cybersecurity regulations by generating detailed detection reports for audits.
5. Research and Development: The system supports researchers in studying malware behavior and developing new detection techniques through its explainability features.

5.ADVANTAGES

The architecture of the Multi-Class Malware Detection System provides several key benefits that enhance detection accuracy, transparency, and usability:

- Enhanced Detection Accuracy: By utilizing modified Graph Neural Networks (GNNs), the system effectively identifies a wide range of malware types, improving overall threat detection capabilities.
- Explainability and Transparency: The incorporation of Explainable AI techniques, like Grad-CAM, helps users comprehend the model's decision-making process, promoting trust and facilitating faster threat response.
- Robust Data Handling: The system processes both byte and ASM files, converting them into image formats for effective analysis, making it adaptable to various data types.
- Scalability: The architecture is designed to handle both small and large datasets efficiently, making it suitable for organizations of varying sizes and cybersecurity needs.
- Support for Cybersecurity Research: The system provides valuable insights into malware behavior, assisting researchers and organizations in developing more effective security solutions.

6. DISADVANTAGES

While the Multi-Class Malware Detection System offers several advantages, it also presents certain challenges and limitations:

- Complexity of Implementation: The integration of Graph Neural Networks (GNNs) and Explainable AI techniques introduces a level of complexity that can make development and deployment more challenging. Ensuring seamless operation requires careful planning and expertise.
- High Computational Demands: The deep learning models used in the system require significant computational resources, leading to increased hardware costs and longer training times, particularly when processing large datasets.
- Dependency on Data Quality: The system's effectiveness heavily depends on the quality and diversity of the training data. Inaccurate or biased datasets can lead to poor detection performance, allowing certain malware variants to evade detection.
- Latency Issues in Real-Time Processing: The complexity of the models and data transformations can introduce latency, potentially affecting real-time malware detection and response.
- Interpretability for Non-Technical Users: While Explainable AI enhances transparency, interpreting the model's reasoning can still be difficult for users without a technical background, which may lead to confusion when analyzing classification results.

7.RESULTS

Binary File Visualization:

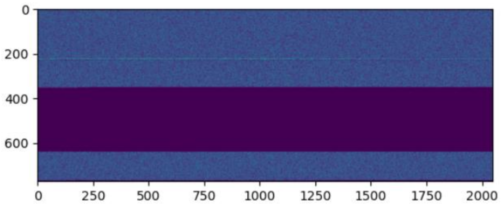

Fig 8.1

Figure 8.1 represents a visualization of binary files, where hexadecimal values from raw byte sequences are converted into grayscale images. The process involves restructuring the byte data into a 2D matrix format, which is then rendered as an image. This technique helps in identifying patterns in malware binaries by treating them as images.

Dataset Image Samples

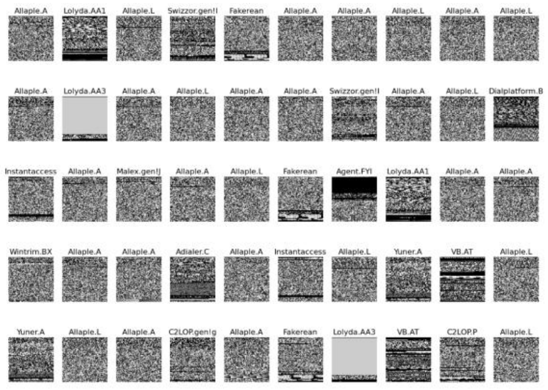

Fig 8.2

Figure 8.2, showcases multiple image samples from the dataset. It is plotted in a grid format with 10 columns and multiple rows, displaying 50 images. The images are classified into different malware categories based on their byte-code structure. This visualization helps in understanding the diversity of data used in training.

Class Distribution in the Dataset

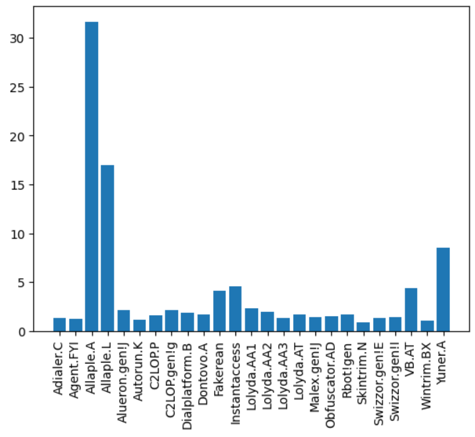

Fig 8.3

Figure 8.3 represents bar chart which shows the percentage distribution of various malware classes in the dataset. Each bar represents a specific malware type, and the y-axis denotes its proportion in the dataset. This visualization is crucial for analyzing class imbalance, which can affect model performance.

Heatmap

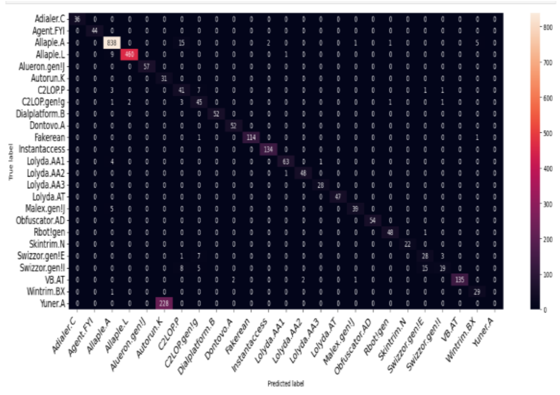

Fig 8.4

Figure 8.4 visualizes the heatmap, which represents the performance of the malware detection model. Each cell in the matrix represents the frequency of correct or incorrect predictions for a specific malware class. Diagonal values correspond to accurate predictions, whereas off-diagonal values indicate misclassifications.

Sample Malware Prediction

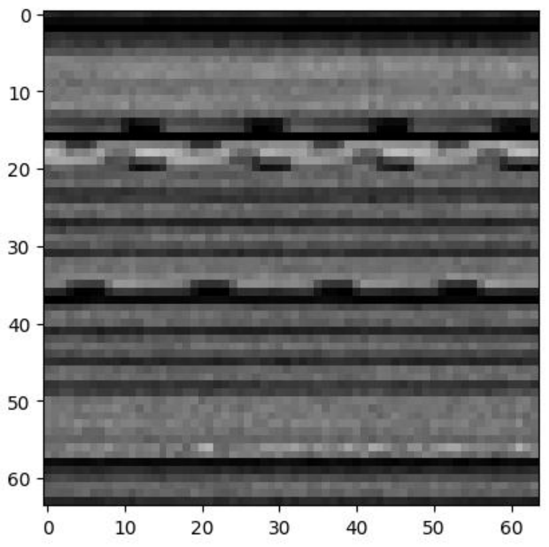

Fig 8.5

Figure 8.5 displays an image of a malware file and its corresponding predicted category. The image is resized and processed before being input to the deep learning model. The output classification label is displayed alongside the image, providing insights into the model’s decision-making process.

Performance Metrics of the Model
Precision: 0.8852
Recall: 0.8764
F1-score: 0.8807
Accuracy (AUC as a proxy): 0.89

8. CONCLUSION

The Multi-Class Malware Detection System, built using modified Graph Neural Networks (GNNs) and Explainable AI, represents a major step forward in addressing evolving cybersecurity threats. By employing advanced machine learning methods, the system improves detection accuracy and efficiency, enabling precise identification of different malware types.

The integration of Explainable AI provides critical insights into the model's decision-making process, fostering user trust and enabling a clear understanding of classification results. This transparency is particularly valuable for cybersecurity professionals, allowing them to respond swiftly and decisively to threats.

Additionally, the system's reliance on high-quality data and sophisticated algorithms ensures adaptability to the ever-changing malware landscape. Continuous refinement and updates will be essential to maintaining its effectiveness against emerging threats.

Overall, the Multi-Class Malware Detection System not only strengthens malware detection capabilities but also deepens understanding of malware behavior, equipping organizations with the necessary tools to enhance their cybersecurity defenses.

APPENDIX

A. Acronyms and Abbreviations

- AI – Artificial Intelligence
- ASM – Assembly Code
- DoH – DNS over HTTPS
- GNN – Graph Neural Network
- ILP – Inductive Logic Programming
- IoT – Internet of Things
- ML – Machine Learning
- PILP – Probabilistic Inductive Logic Programming
- SSL – Self-Supervised Learning
- UI – User Interface

B. Tools and Technologies Used

- Programming Languages: Python, TensorFlow, PyTorch
- Machine Learning Libraries: Scikit-learn, Keras, NetworkX
- Development Tools: Jupyter Notebook, VS Code
- Data Processing Tools: Pandas, NumPy
- Visualization Tools: Matplotlib, SHAP (SHapley Additive exPlanations)

C. Datasets Used

- ASM Dataset: Contains structural features extracted from assembly code of executable files.
- Merged Dataset: Combination of ASM and Byte datasets for enhanced malware classification.

D. System Requirements

- Hardware Requirements:
  - Minimum: 8GB RAM, 4-core CPU, 50GB storage
  - Recommended: 16GB RAM, GPU support (NVIDIA), 100GB storage
- Software Requirements:
  - OS: Windows 10, Linux (Ubuntu)
  - Python 3.8+
  - Required Libraries: TensorFlow, PyTorch, Scikit-learn, Pandas

REFERENCES

[1] Z. Zhang, L. Yilmaz and B. Liu, "A Critical Review of Inductive Logic Programming Techniques for Explainable AI," in IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 8, pp. 10220-10236, Aug. 2024, doi: 10.1109/TNNLS.2023.3246980. https://ieeexplore.ieee.org/document/10092808

[2] D. Vlahek and D. Mongus, "An Efficient Iterative Approach to Explainable Feature Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 5, pp. 2606-2618, May 2023, doi: 10.1109/TNNLS.2021.3107049. https://ieeexplore.ieee.org/abstract/document/9528915

[3] T. Zebin, S. Rezvy and Y. Luo, "An Explainable AI-Based Intrusion Detection System for DNS Over HTTPS (DoH) Attacks," in IEEE Transactions on Information Forensics and Security, vol. 17, pp. 2339 2349, 2022, doi: 10.1109/TIFS.2022.3183390. https://ieeexplore.ieee.org/document/9796558

[4] S. Ali, O. Abusabha, F. Ali, M. Imran and T. Abuhmed, "Effective Multitask Deep Learning for IoT Malware Detection and Identification Using Behavioral Traffic Analysis," in IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 1199-1209,

June 2023, doi: 10.1109/TNSM.2022.3200741. https://ieeexplore.ieee.org/document/9865131

[5] S. Lai, W. Luan and J. Tao, "Explore Your Network in Minutes: A Rapid Prototyping Toolkit for Understanding Neural Networks with Visual Analytics," in IEEE Transactions on Visualization and Computer Graphics, vol. 30, no. 1, pp. 683-693, Jan. 2024, doi: 10.1109/TVCG.2023.3326575. https://ieeexplore.ieee.org/document/10308640

[6] B. Esmaeili, A. Azmoodeh, A. Dehghantanha, H. Karimipour, B. Zolfaghari and M. Hammoudeh, "IIoT Deep Malware Threat Hunting: From Adversarial Example Detection to Adversarial Scenario Detection," in IEEE Transactions on Industrial Informatics, vol. 18, no. 12, pp. 8477-8486, Dec. 2022, doi: 10.1109/TII.2022.3167672. https://ieeexplore.ieee.org/document/9760120

[7] D. Trizna, L. Demetrio, B. Biggio and F. Roli, "Nebula: Self-Attention for Dynamic Malware Analysis," in IEEE Transactions on Information Forensics and Security, vol. 19, pp. 6155-6167, 2024, doi: 10.1109/TIFS.2024.3409083. https://ieeexplore.ieee.org/document/10551436

[8] S. Zillien and S. Wendzel, "Weaknesses of Popular and Recent Covert Channel Detection Methods and a Remedy," in IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 6, pp. 5156-5167, Nov. Dec. 2023, doi: 10.1109/TDSC.2023.3241451. https://ieeexplore.ieee.org/document/10034794