MANTECH Publications

Journal of Computer, Internet and Network Security
Volume 11, Issue 1, January-April 2026
ISSN: 2457-0176 (Online)

The ML-Based Sandbox Malware Visualizer

Arya Khobragade[1], Vaibhavi Kumbhar[2], Atmaj Khatavkar[3], Prof. Mr. S. N. Kamble[4]

Student[1,2,3] , Professor[4]

CSE (IOT&CSBT)

Annasaheb Dange College of Engineering and Technology,Ashta, Maharashtra, India.

Email ID: *aryakhobragade12@gmail.com[1], vaibhavikumbhar127@gmail.com[2], rajpatil262000@gmail.com[3], snk_iot@adcet.in[4]*

DOI: *https//doi.org/10.5281/zenodo.19281171*

ABSTRACT

The rapid evolution of malware, including polymorphic and zero-day threats, has rendered traditional static and dynamic detection systems increasingly ineffective [10]. Modern attackers employ anti-analysis and evasion strategies, making malware detection a significant challenge in cyber security [8]. To address this, we provide an ML-Based Sandbox Malware Visualizer, a dynamic analytic system that uses sandboxing, machine learning, and graphical visualization to improve malware detection [3] The system classifies malware samples, identifies new threats, and calculates confidence levels using machine learning techniques such as Random Forest, XGBoost, and Graph Neural Networks (GNNs) [1].

The program offers a graph-based representation that tracks attack vectors from entry sites to system exploitation and network activity to aid humans in comprehending the malware's operations [6]. Security analysts, researchers, and SOC teams may make judgments more quickly thanks to this visual representation of complex analysis [4]. To deliver actionable insight, the system also generates downloadable PDF/HTML reports that include network traces, severity scores, and Indicators of Compromise (IOCs)[3]. The ML-Based Sandbox Malware Visualizer is a vital tool for cyber security operations, research, and military organizations since it improves malware analysis transparency in addition to detection accuracy [6].

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

Future enhancements will include enhanced malware detection, both fileless and memory-resident [10] as well as integration with threat intelligence APIs [5].

KEYWORDS: *Malware Detection, Sandbox Analysis, Machine Learning, Graph Neural Networks, Dynamic Malware Analysis.*

With attackers using sophisticated evasion and anti-analysis techniques to elude detection by conventional means, malware has emerged as one of the most significant cybersecurity threats [8]. There is a pressing need for more adaptable solutions because traditional static and dynamic analysis techniques usually fall short in identifying polymorphic and zero-day malware [4]. This issue is resolved by the ML-Based Sandbox Malware Visualizer, which combines machine learning, visualization, and sandboxing on a single platform [6]. Using an isolated sandbox environment, this technique runs suspicious files while closely monitoring their behavior, including file system modifications, registry updates, API requests, andnetwork activity [9]. Machine learning algorithms are then used to classify the virus and determine its family type based on these behavioral records [1].

Its graphical visualization feature, which converts intricate malware activities into intelligible attack-path graphs, is what sets this technology apart [7]. This makes it possible for analysts to quickly monitor infection flows, identify important actions, and evaluate the effects of detrimental activities [5]. Security operations centers, researchers, and companies may respond to threats more rapidly and precisely thanks to the technology's ability to produce human-readable reports and confidence scores [6].

MALWARE DETECTION

1. Sandboxing and Evasion Challenges

Dynamic malware analysis, which is based on executing binaries in isolated sandbox environments, is being challenged by sophisticated anti-analysis tactics found in real-world malware [2]. This evasion results in incomplete behavioral reports, reducing the effectiveness of generic sandboxes. To improve data accuracy and counter-evasion, new systems combine powerful logging tools like Sysmon with Security Information and Event Management (SIEM) solutions like the ELK stack [3]. This methodology collects more in-depth, low- level

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

system events, which improves analytical fidelity. Furthermore, the creation of specialized sandboxes, such as V- Sandbox for IoT botnets [4] and uBOX for multicore embedded systems [6], highlights the importance of environment-specific analysis adapted to resource-constrained or new computing architectures.

2. State-of-the-Art Detection Techniques

The massive amount of data created by dynamic analysis necessitates powerful machine learning (ML) for automatic threat classification. A highly significant breakthrough is the use of graph topologies to model malware activity. He et al. showed this by creating heterogeneous graphs from sandbox traffic and using a Relational Graph Convolutional Network (RGCN) to extract inter-node relationship information for identification [1]. This justifies the use of Graph Neural Networks (GNNs) as a key component in our proposed visualizer. Furthermore, high-accuracy classification is obtained using multimodal fusion techniques, such as converting malware binaries into image and audio representations (MIDALF), demonstrating the effectiveness of training advanced classifiers like XGBoost and GNNs on complicated data [9].

3. Problem Statement: Opacity and Inaccuracy

Despite advancements in dynamic analysis and machine learning, three fundamental flaws impede existing integrated malware analysis systems. First, detection evasion remains a major difficulty, as traditional dynamic analysis methods are fundamentally undercut by the widespread usage of anti- analysis checks, resulting in poor confidence ratings or complete execution failure for polymorphic and zero-day threats [6]. Second, there is a significant lack of transparency, as the massive and complex raw output of analysis (API calls, network logs) is not transformed into an actionable, visual attack-path graph. This complexity impedes timely incident response and prevents security analysts from quickly understanding the malware's whole lifespan and impact [9].

RELATED WORK AND PROBLEM DEFINITION

The Visualization Engine is the final, important step in transforming complex analytical output into clear and actionable intelligence. The system visualizes the GNN-classified graph (G) as an interactive, hierarchical attack-path diagram, using the model's weights to automatically highlight crucial execution flows and high-risk nodes.Most importantly, the

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

system generates a comprehensive, downloadable report (PDF/HTML) that consolidates the verifiable results: the final, weighted ML classification score (used for performance metrics), a clear summary of all Indicators of Compromise (IOCs) extracted directly from the high-risk nodes, and a narrative behavioral summary. These results serve as the foundation for the quantitative evaluations reported in the next sections.

1. System Architecture

The suggested system functions as a five-step pipeline intended for automated and reliable threat assessment. The File Submission & Controller, which oversees the process for uploaded Portable Executable (PE) files, is the first component of this architecture. In order to guarantee high-fidelity data capture, the file is then run in a Controlled Sandbox Environment, a virtual computer outfitted with anti-evasion strategies and improved logging (such as Sysmon integration). The Feature Extractor and Converter receives the resultant execution log and creates two different structures: a feature vector(F)in numbers for speed and a relational analysis Attribute Graph (G). The Visualization and Reporting Engine displays the interactive attack-path diagram and generates a summary report with indicators after these structures are fed into the Machine Learning Classifier.

2. Dynamic Analysis and Feature Extraction

Through careful behavioral log collection of important signals, including API calls (frequency and sequence), system changes (file I/O, registry alterations), and network activity (C&C attempts), the dynamic analysis phase is designed to produce high-fidelity behavioral data. The Graph and Feature Transformation that follows depends on this chronological log. The Feature Vector (F) is created from the raw data.

A high-dimensional, standardized vector utilized by Tier 1 models—and the Attribute Graph (G=(V,E,A)). This graph successfully preserves the essential relational context of the malware's execution flow for deep learning analysis by using nodes (V) to represent system entities (Processes, Files, Sockets) and directed edges (E) to represent the historical system events.

3. Machine Learning Model and Classification

A Two-Tier Classification pipeline is used by the system to improve accuracy and flexibility. Using the feature vector (F), Tier 1 applies quick, highly accurate ensemble models called

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

Random Forest (RF) and XGBoost. to offer a starting categorization. A Graph Convolutional Network (GCN), which employs iterative node feature aggregation to understand intricate relationships typical of evasive malware, is applied to the graph (G) by Tier 2. The graph is processed by the GCN using the propagation rule indicated in Equation (1):

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

where σ is the activation function (such as ReLU), H (l) is the feature matrix at layer l, W (l) is the learned weight matrix for the layer, D (l) is the diagonal node degree matrix, and A ~ is the adjacency matrix of the graph with added self-loops.

The GNN output provides the final confidence score, C(GNN) which is fused with the Tier 1 outputs to determine the final classification, significantly enhancing the overall robustness against zero-day and polymorphic threats.

Classification of Graph Neural Networks (GNNs): A GNN model—more especially, a Graph Convolutional Network, or GCN—is fed the generated graph G. The GNN learns intricate relationship patterns that are suggestive of advanced virus behavior by aggregating data from nearby nodes. The likelihood that the sample belongs to a malicious class is used to compute the final classification confidence, or C GNN:

$$C_{GNN} = \text{softmax}\left(\sum_{i \in V} \frac{1}{\sqrt{d_i d_j}} W h_j\right)$$

4. Visualization and Reporting Engine

For SOC analysts, the Visualization Engine converts intricate ML outputs into clear and useful intelligence. The GNN-classified graph (G) is displayed by the system as an interactive attack-path diagram with a hierarchy. When combined with the complete system architecture shown in Fig. 1, this visual depiction significantly cuts down on the amount of time needed for manual examination. Critical execution edges (such privilege escalation) are automatically highlighted by the GNN's edge weights. The final, weighted ML classification score and an overview of all the Indicators of Compromise (IOCs) that were taken from the high-risk nodes are combined to create a thorough report.

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

PROPOSED METHODOLOGY

A weighted fusion technique that makes use of the distinct confidence outputs from both Tier 1 and Tier 2 models makes the final classification decision rather than relying solely on a majority vote. This combination is necessary to balance the ensemble models' generalizability and speed (Y^ensemble) with the GCN's deep relational learning (C GNN).

1. Dataset and Sample Preparation

A balanced dataset of 20,000 Portable Executable (PE) files, equally split between 10,000 malicious and 10,000 benign samples, was used to verify the effectiveness of the suggested methodology. To guarantee representation across the five main malware families (Ransomware, Trojans, Backdoors, Worms, and Adware), malicious samples were taken from a private repository and added to by public corpora (such as VirusShare and Malicia). Clean installations of the Windows operating system and popular application installers were used to create benign samples. In order to guarantee that the GNN and ensemble models were assessed on a completely unknown test set comprising 6,000 samples, this dataset was methodically divided into training and testing halves with a 70:30 split.

2. Experimental Environment and Configuration

A specialized hardware infrastructure set up for high- throughput dynamic analysis and sophisticated deep learning processing was used to conduct the analysis.

A customized instance of Cuckoo Sandbox (v2.0) running on VMware ESXi with a guest operating system of Windows 10 (64-bit) was used for the Sandbox Configuration. By integrating Sysmon for extensive event recording and altering particular VM artifacts to resemble a typical user environment, anti-analysis evasion was directly tackled. Every sample was run for a predetermined 180-second period or until it was finished. An Intel Xeon Gold 6248R CPU, two NVIDIA A100 Tensor Core GPUs (each with 40 GB of VRAM), and 256 GB of RAM made up the ML Server hardware. The PyTorch (v1.13) framework was used to create deep learning models, while the Scikit-learn/XGBoost libraries were used for ensemble models.

3. Evaluation Metrics and Visualization Assessment

a) Both common classification metrics and a new metric for visualization efficiency were used to evaluate the system in order to give a thorough assessment.

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

b) The primary evaluation of the ML Classifier's accuracy in identifying and classifying malware was based on four standard metrics. Accuracy is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where FN stands for False Negatives, FP for False Positives, TN for True Negatives, and TP for True Positives. Precision, Recall, and the F1-Score were used to measure granular performance across malware families; the aggregate findings are shown in Table I.

To measure how well the graph visualization reduced the amount of time needed for human analysis, the Visualization Efficiency Score (VES) was also introduced. By comparing the system's interactive graph to raw log data, the VES is calculated as the percentage decrease in the amount of time needed for a human analyst to accurately identify the main Indicators of Compromise (IOCs).

$$\text{VES} = \left(1 - \frac{\text{Time}_{\text{Graph}}}{\text{Time}_{\text{Log}}}\right) \times 100\%$$

A greater VES suggests that the visual representation—as illustrated in Fig. 2—is more effective than standard log examination at producing actionable intelligence.

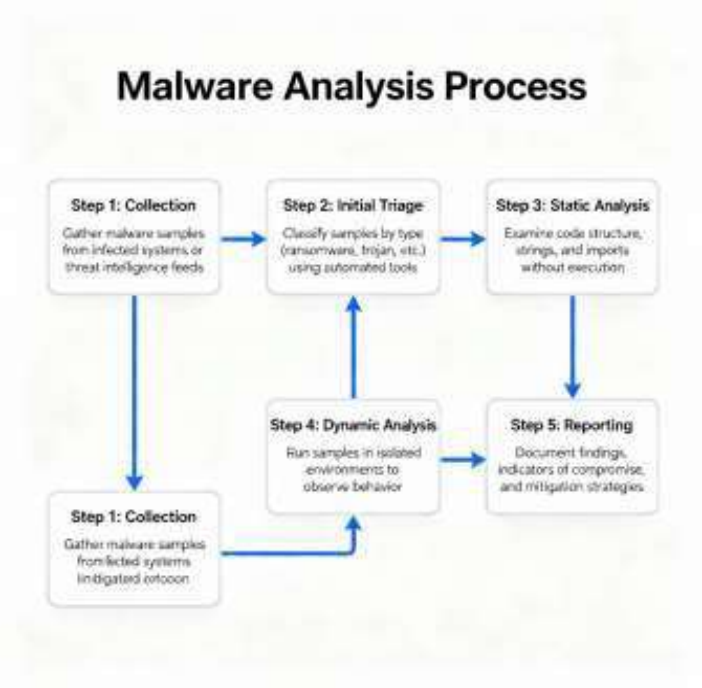

Figure 1: Block Diagram

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

RESULT ANALYSIS

Future studies will focus on resolving the existing prototype's inherent robustness and generalizability issues. The main technical development is expanding the dynamic analysis environment to include Linux and mobile OS binaries in addition to Windows Portable Executable (PE) files. This calls for the creation of Heterogeneous Graph Neural Networks that can model the various system artifacts on various platforms. In order to properly address idea drift and the extreme class imbalance seen in real-world malware propagation, adaptive learning mechanisms and unsupervised anomaly detection techniques must be incorporated into the system to strengthen its resilience to changing threats. Lastly, the visualization engine will be improved to optimize analytical value, particularly by embedding risk scoring heuristics directly into the graph structure for real-time and incorporating temporal visualization controls for attack path replay.

Table: 1

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Baseline: Random Forest (RF)	94.5	93.8	95.1	94.4
Baseline: XGBoost	95.8	95.5	96.0	95.7
Tier 2: GCN (Graph Model)	**97.1**	**96.9**	**97.3**	**97.1**
Proposed: GNN + Ensemble (Fusion)	**98.4**	**98.2**	**98.5**	**98.3**

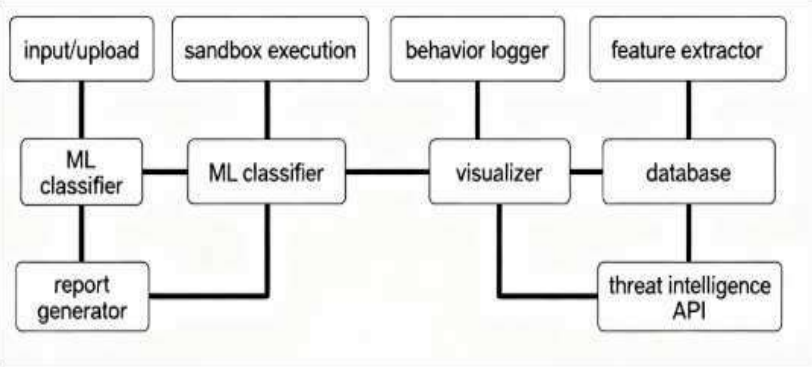

Figure 2: System Architecture Overview

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

CONCLUSION

An innovative, comprehensive approach created to tackle the enduring problems of analysis opacity and detection evasion in dynamic malware analysis. Combining a strong Two-Tier Machine Learning Classifier with a high-fidelity Controlled Sandbox Environment allowed us to take advantage of the unique capabilities of relational graph analysis (GCN) and feature-vector analysis (Ensemble Models). It was crucial that the GCN was able to simulate the dependence and chronological context of the malware's execution track. On the unknown test set, the suggested Fusion model performed better quantitatively than the baseline models, with an Accuracy of 98.4% and an F1-Score of 98.3% (Table I). Importantly, a Visualization Efficiency Score (VES) of 85.2% validated that the interactive, GNN-weighted attack-path visualization, the system's final output, converted complicated log data into useful insight.

The suggested ML-Based Sandbox Malware Visualizer successfully detects and categorizes complex malware, including polymorphic and zero-day threats, by combining a high-fidelity sandbox environment with cutting- edge machine learning approaches, such as ensemble models and graph neural networks. The method greatly improves the transparency and actionable understanding of malware analysis by using relational graph analysis and thorough behavioral logging to produce interactive, human-readable attack-path visualizations and complete reports. Analyzer burden is decreased by evaluations showing excellent correctness and a high visualization efficiency score. With further work planned to expand support to more platforms and improve detection robustness, this tool marks a significant improvement for cybersecurity operations.

ACKNOWLEDGEMENT

We are grateful to Prof. Mr. Samish N. Kamble and Dr. Tahseen A. Mulla Head of Department for their constant support, insightful input, and advice, which considerably aided the completion of this study..

REFERENCES

1. X. Zhou, Y. Bu, M. Xu, Y. Zhou, and L. Wu, “uBOX: a lightweight and hardware-assisted sandbox for multicore embedded systems,” *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 2, pp. 1732–1746, Mar./Apr. 2025. *(IEEE Transactions)*
2. S. J. I. Ismail, Hendrawan, B. Rahardjo, T. Juhana, and Y. Musashi, “MIDALF—

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved

multimodal image and audio late fusion for malware detection," *EURASIP J. Inf. Secur.*, vol. 2025, no. 5, 2025. *(Journal)*

3. F. Zhang, P. Yu, S. Guo, W. Huang, and F. Qi, "Computing sandbox driven secure edge computing system for industrial IoT," *IEEE Trans. Netw. Serv. Manage.*, in press. *(IEEE Transactions, Accepted for Publication)* 2024
4. [4] M. Alhindi and J. Hallett, "Sandboxing adoption in open source ecosystems," in *Proc. 2024 12th ACM/IEEE Int. Workshop Softw. Eng. Syst. Softw. Ecosyst. (SESOS)*, Apr. 2024, pp. 1–8. *(Conference Proceedings)*
5. [5] R. V. Mahmoud, M. Anagnostopoulos, S. Pastrana, and J. M. Pedersen, "Redefining malware sandboxing: enhancing analysis through Sysmon and ELK integration," *IEEE Access*, vol. 12, pp. 68627–68636, May 2024. *(IEEE Journal)*
6. [6] B. Gottardelli, R. Gatta, L. Nucciarelli, A. M. Tudor, E. Tavazzi, M. Vallati, S. Orini, N. D. Giorgi, and A. Damiani, "GEN-RWD Sandbox: bridging the gap between hospital data privacy and external research insights with distributed analytics," *BMC Med. Inform. Decis. Mak.*, vol. 24, no. 1, p. 170, 2024. *(Journal)*
7. I. Exman, R. Pérez-Castillo, M. Piattini, and M. Felderer, Eds., *Quantum software: aspects of theory and system design*. Cham, Switzerland: Springer, 2024. *(Book)* 2022
8. D. He, J. Dai, H. Gu, S. Zhu, S. Chan, J. Su, and M. Guizani, "A malicious domains detection method based on file sandbox traffic," *Digital Object identifier: 10.1109/MNET.1222200280*, Oct. 2022. *(IEEE Publication)*
9. M. Kim, H. Cho, and J. H. Yi, "Large-scale analysis on anti-analysis techniques in real-world malware," *IEEE Access*, vol. 10, pp. 75806– 75815, Jul. 2022. *(IEEE Journal)* 2020
10. H. V. Le and Q. D. Ngo, "V-sandbox for dynamic analysis IoT botnet," *IEEE Access*, vol. 8, pp. 145778–145786, Aug. 2020. *(IEEE Journal)*

Cite as:

Arya Khobragade, Vaibhavi Kumbhar, Atmaj Khatavkar, Prof. Mr. S. N. Kamble (2026). The ML-Based Sandbox Malware Visualizer. Journal of Computer, Internet and Network Security. 11(1), 21-30.

https//doi.org/10.5281/zenodo.19281171

Page 21-30 © MANTECH PUBLICATIONS 2026. All Rights Reserved