MULTIDISCIPLINARY RESEARCH INSTITUTE INDIA

Open Access International Journal of Science and Engineering

ISSN: 2456-3293

Volume 9 Issue 1s (2026) | Pages: 50-56

LLM-Assisted Cyber Threat Analysis Using Static Features and Malware Pattern Mining: A Comprehensive Review and Future Directions

Tamanna Jethani[1], Sneha Malhotra[2]

[1,2]Dept of Computer Science and Business System, Bharati Vidyapeeth Deemed University College of Engineering, Pune,India

[1]*tamannajethani@gmail.com*, [2]*snehamalhotra843@gmail.com*

Peer Review Information

***Type**: Article*
***Received**: 29 March 2026*
***Revised**: 26 April 2026*
***Accepted**: 30 May 2026*
***Published**: 19 June 2026*

Abstract

Cybersecurity is one of the most significant global challenges due to rapid growth of ransomware and polymorphic malware. Traditional detection methods, such as signature-based and static analysis techniques, are effective only for known threats and often fail against zero-day attacks and obfuscated malware. The paper focuses on behavior-based malware detection, which enables real time analysis and improved detection of unknown malware. The proposed system employs a hybrid approach that combines dynamic behavioral indicators with AI-driven prompt engineering which used LLM and ML hybrid model. Experiments were conducted using dataset including [Malware Analysis Datasets: API Call sequences, Malware Analytics Dataset Leveraging Cuckoo Sandbox, Malware detection model, CICIDS-2017], and performance evaluated using standard metrics. The results demonstrate that the proposed approach achieves more accuracy, outperforming traditional models, highlighting its effectiveness for early-stage malware detection.

Keywords: Malware Detection; Behavioral Analysis; Cybersecurity; Real-Time Threat Detection; Machine Learning; Early Threat Detection; Hybrid ML-LLM Framework.

How to Cite This Article

Jethani, T., & Malhotra, S. (2026). LLM-Assisted Cyber Threat Analysis Using Static Features and Malware Pattern Mining: A Comprehensive Review and Future Directions. *Open Access International Journal of Science and Engineering*, 9(1s), 84-90.

Introduction

Cybersecurity is one amongst the major concerns of today's era due to exponential growth of malware, specifically ransomware and polymorphic threats. According to AV-Test (2024), approximately 560,000 new malwares are found daily [1]. Systems that were used traditionally are unable to handle evolving malwares and their variants. The influx of new samples is so fast that it is difficult to detect them on time and capture signatures, which motivates the move beyond signature-based detection. This includes fileless malware, living-off-the-land binaries (LOLBins), and heavily obfuscated malware, demanding more advanced real-time detection mechanisms.

Conventional malware detection methods include signature-based and static analysis techniques. Signature-based detection relies heavily on predefined patterns and is effective in detecting only already known and fingerprinted malware. Zero-day malware and polymorphic variants change their appearance to hide those fingerprints, which results in high miss rates. Static analysis examines files without running them and can be blinded by packers or obfuscation, often missing the real intent that only appears during execution [2].

Dynamic analysis methods, although more robust, are performed in offline sandbox environments, making them unsuitable for real-time detection.

To overcome these limitations, behavior-based malware detection has gained significant importance in recent years. Instead of analysing static code signatures, behavior-based approaches monitor reliable behavioral indicators that extract API calls, file operations, registry changes, network traffic, memory or process behavior, and system log anomalies. These indicators help provide valuable insight regarding malicious content and unknown variants and are much harder for malware authors to obfuscate [3].

Recent advancements in Machine Learning and Large Language Models (LLMs) have opened new opportunities for intelligent analysis of malware. Machine learning classifiers can be used to process behavioral features, whereas LLMs can provide better contextual understanding through prompt-driven analysis of execution logs and behavioral traces. They can be used to continuously capture and analyse behavioral events for immediate threat detection. Despite their potential, they are used independently in most studies, and there is a lack of optimized hybrid approaches, leading to models that lack robustness, scalability, or interpretability.

In this paper, we propose a real-time behavior-based malware detection framework designed for real-time detection. It combines dynamic behavioral indicators with a hybrid ML and LLM approach. The system continuously monitors runtime behavior, extracts critical features, verifies malicious intent, and performs early-stage threat detection. It enhances accuracy in ambiguous or borderline cases, improving zero-day malware detection capacity.

The contributions through this are as follows:

- Design a real-time behavioral monitoring pipeline for early detection of malware.
- Integration of ML and LLM-assisted prompt engineering for enhanced accuracy.
- Evaluation using multiple dataset demonstrating superior performance than traditional system.
- Significantly improved accuracy and reduced false positive in early stage detection.

Literature Survey

Recent studies have been conducted for malware and ransomware detection using ML and LLMs in order to improve detection accuracy, understanding, and decision-making. These approaches rely mainly on static analysis, dynamic sandbox analysis, or a combination of both.

BEACON (2025) proposes a framework to analyze sandbox execution logs. It uses a combined approach comprising embeddings generated by LLMs with Convolutional Neural Networks (CNNs). The model captures semantic modelling of malware behavior, helping to increase detection accuracy for new malware variants. However, dependency on sandbox environments and deep neural networking leads to computational overhead, making the system slow for real-time detection [4].

MalAware (2025) uses LLMs on sandbox environments for summarization of extracted malware behavior. It helps in enhancing explainability and understanding. Although the method is effective, it does not support pre-encryption detection and works only on post-encryption detection [5].

The Temporal Graph-Based Detection Model (2025) helps in capturing successive and dependency-based relationships among system events in order to detect ransomware behavior and actions. The model efficiently represents complex temporal patterns but has a high computational cost and, due to latency limitations, is not applicable for real-time analysis [6].

A 2024 review based on the analysis of LLMs in malware analysis highlighted prompt engineering techniques and associated risks. However, the study provides no information about real-time detection and detection time requirements, which limits its operational importance [7].

Between 2023 and 2024, surveys were conducted on ML techniques for early-stage ransomware detection, and the results demonstrated feasibility prior to encryption. However, limited datasets and high false-positive rates make these approaches less reliable for practical deployment [8].

The above studies present several major limitations:

- Heavy reliance on sandbox environments.
- Lack of hybrid systems combining ML and LLM synergy.
- Limited datasets and real-time behavioral analysis.
- Slow inference time for deep learning models.
- Poor interpretability in ML-only approaches.

These limitations highlight the need to develop a real-time, hybrid ML-LLM behavior-based detection framework that can improve accuracy while reducing false positives and latency.

Methodology

The presented system adopts a hybrid approach for detection of malware that integrates both Machine learning (ML) and Large Language Model (LLM)-driven contextual analysis. The methodology is designed to continuously monitor system behavior, extract relevant features, and perform multi-stage classification in real time . The complete workflow is illustrated through the following subsections.

System Architecture Overview

The suggested architecture contains of four major components:

- Behavioral Data Collection Module.
- Feature Extraction and Preprocessing Module.
- Machine learning-based Detection Engine.
- LLM-Assisted Contextual Verification Layer.

The initial information gathering occurs in the form of runtime behavioral events in programs under monitoring as earlier described in [2], [3] . The gathered information in the form of extracted features is analysed with ML classifiers for initial prediction purposes. However, in such uncertain conditions with low confidence levels,

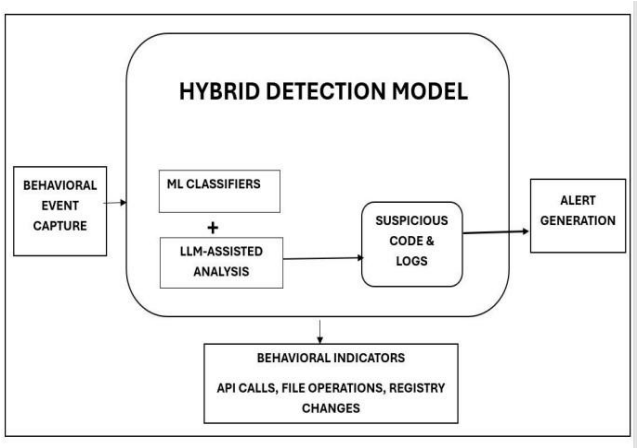

Fig. 1. *Proposed hybrid ML-LLM detection architecture*

Behavioral Data Collection

It collects behavioral data from various dynamic analysis sources, such as API call sequences, registry modifications, file system activities, network traffic, and process-level anomalies. The datasets used include the API Call Malware Dataset, Cuckoo Sandbox Logs, Malware Behavior Analytics Dataset, and CICIDS-2017, which are widely utilized in malware behavior research [4], [6], [8]. Each dataset provides diverse real-world malicious activity patterns.

Key events collected in the behavioral data file include file operations, registry modifications, network communication attempts, process creation patterns, inter-process interactions, and system-level API calls. This module ensures that even heavily obfuscated or polymorphic malware can be captured through its runtime actions instead of static signatures.

Feature Extraction and Preprocessing

The raw logs are complete with redundant and noisy events; Hence, it requires preprocessing. Event normalization is conducted by the

pipelines to standardize the pattern of system calls [7], [8]. Sequence of API calls are encoded using n-gram modeling, TF-IDF representations Other statistical features derived includes call frequency, entropy, file access ratios and characteristics of network packets. Dimensionality reduction and feature selection method are carried out in order to increase efficiency of model and reduce overfitting [8].

Machine Learning-Based Detection Engine

This acts as the main classification layer for Machine Learning (ML). Random Forest, SVM, XGBoost, or LSTM algorithms are used for training with extracted behavioral features [4], [6], [8]. The dataset is split into training and test sets, and models are evaluated using accuracy, precision, recall, and F1-score [4], [7]. Predictions with scores higher than a predefined confidence threshold are directly accepted as final outputs. Predictions with confidence scores below the threshold are fed into the LLM for further contextual processing.

LLM-Assisted Contextual Verification

This enhances the quality of the decision-making process. In the LLM module, the behavior traces received by the system are subjected to a semantically informed evaluation. This is done by using prompt engineering, where the execution logs along with features detected are made available for evaluation using the LLM. This ensures the mitigation of any possible misconceptions, which could have been made using the ML approach [7].

Hybrid Decision-Making Process

Finally, the classification stage is a fusion mechanism, in which the ML prediction coincidence is taken into accounts before the final classification is done via the LLM-defined judgement context. It means if ML has a high confidence level in classification ,the classification can be taken directly from ML, otherwise is done via the LLM judgement[4] ,[5] .

Real Detection Pipeline

This model is also capable of facilitating a real time system by continuously monitoring the system , performing feature extraction, making instant ML predictions , as well as performing validations of the LLM models .Alerting is done in real time as soon as potential malicious activities are recognized .The suggested model is also favorable for the identification of pre-encryption ransomware as well as the timely prevention of system compromise[2] , [3].

Experimental Setup

The proposed malware detection model, which is a hybrid version of the ML and LLM approaches, is implemented and validated in a control environment with several malware behavior-based datasets. The following is an overview of how the proposed model is setup.

System Architecture Overview

All experiments were performed on workstation with following configurations:

- Intel core i5/Ryzen 5 processors
- 16GB RAM
- 512GB SSD storage
- Windows 10/ Ubuntu 20.04 operating system This setup is required for adequate computational capacity of model.

Software environment

The proposed system was implemented using following software components:

- Python 3.10 for backend development and model execution.
- Streamlit for building the real-time detection dashboard.
- Scikit-learn for training ML models
- Pandas and NumPy for data processing and feature alignment.
- Joblib for loading serialized data model artifacts (.pkl files).
- Cuckoo Sandbox, Sysmon and Process Monitor for generating behavioral logs such as API calls, registry activity and file-system events.
- LLM integration through prompt engineering using external LLM APIs for contextual reasoning, consistent with LLM-based analysis approaches reported in [4] ,[5].

Datasets

Four behavioral datasets are used for evaluation:

1. API Call Malware Dataset- benign and malicious API sequences.
2. Cuckoo Sandbox Logs -execution logs containing file, registry, process, and network behavior.
3. Malware Behavior Analytics Dataset-ransomware traces and post infection behavior.
4. CICIDS-2017 -labeled network intrusion traffic. These datasets have been widely used in prior behavioral malware detection

research and studies. The dataset upload interface and automatic behavioral type identification are in Fig. 2.

Fig. 3. *Dataset upload interface and automatic behavioral type identification*

Preprocessing

Raw behavioral logs were processed through:

- Event filtering and normalization
- Removal of redundant or duplicate events
- API sequence encoding using n-grams and TF-IDF
- Extraction of statistical features (frequency, entropy, access ratios)
- Dimensionality reduction to eliminate noise and reduce overfitting.

Preprocessing produced standardized feature vectors suitable for ML and LLM processing.

Training and Testing Procedures

The dataset was split 70% and 30% for training and testing. A generalization of results was achieved by the implementation of a 5-fold cross validation. Particiapting ML models were chosen to be Random Forest, SVM, XGBoost,and LSTM based on traditionally followed practices in past research articles related to malware detection Low confidence ML model predictions, which were below a particular threshold, were further subjected to verification through the use of LLM.

Evaluation Metrics

Performance assessment using standard cybersecurity evaluation metrics [7], [8]:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion metrics
- False positive Rate (FPR)
- Detection Latency

This allows estimation of both effectiveness and real-time applicability of the proposed framework.

Results and Discussion

The presented detection framework was tested with several behavioral datasets, demonstrating the effectiveness of the proposed approach on improving the detection accuracy.

Machine Learning Model Performance

In offline evaluation, the ML models are trained to operate on static, dynamic and Cuckoo Sandbox behavior features. The model's performance is measured based on Accuracy, Precision, Recall, F1-score, False Positive Rate (FPR), Confusion Matrix etc.

Across all datasets: The highest performance was recorded in case of Random Forest. XGBoost performed well with dynamic API call data. LSTM demonstrated better understanding of the sequences but also required more computation. Thus, this hybrid framework helped minimize false positive results in borderline behavioral samples where ML confidence is low. A sample ML prediction output generated by system is shown in Fig. 3.

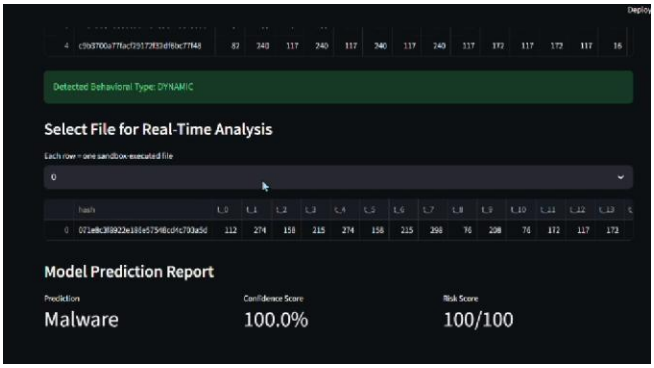

Fig. 3. *Machine Learning prediction panel displaying verdict, confidence score, and risk level.*

Impact of LLM-Assisted Verification

Usage of LLM module was focused on the reinterpretation of low-confidence predictions generated in the process of machine learning, tracing behavioral characteristic and sequence of events . This greatly increased the reliability pertaining to detection :

1. Enhanced Classification of Ambiguous API Sequences.
2. Better identification of early-stage ransomware behavior
3. Enhanced reasoning power lacking in ML system alone
4. The LLM also gave a human-readable explanation, thus increasing the interpretability and trustworthiness of detection process

An example of LLM-based contextual reasoning for threat interpretation is shown in Fig. 4.

Fig. 4. *Prompt Engineering used to generate LLM-based contextual reasoning*

Real-Time Detection Results

In the real-time Streamlit interface ,the system generated:

- Malware/Benign verdict
- Confidence score
- Threat level (High/Medium/Low)
- Risk scoring
- LLM reasoning output

The pipeline's execution of each behavioral log was found to take less than 1-2 seconds , which confirmed the pipeline's feasibility in a real-time execution environments. The final Real-time detection output generated by ML-LLM-system is represented in Fig. 5.

Fig. 5. *Real-time detection output showing LLM-based threat level, recommended action and explanation*

Comparison With Existing Approaches

Compared to prior work [4], [6], [8]:

- The hybrid ML-LLM approach gives lower false positives.
- Additionally, t h e system was not dependent exclusively on sandbox execution.
- Automatically identifies behavioral indicators than the ransomware encryptions.
- Allows for more easily interpretable reasoning compared to the exclusive property of deep.

Conclusion

In the presented paper, an integrated hybrid behavior-based malware detection system, combining machine learning-based classifiers along with the power of Large Language Models for better accuracy of threat detection results, is developed. The proposed system includes the collection of detection behaviors with the assistance of dynamic, static, and sandbox-generated log-based behaviors, followed by feature engineering for better interpretation of the results and multi-stage classification. The ML component is used for fast prediction, while the LLM refines low-confidence outputs by generating contextual explanations and threat assessments.

Experimental results show reduced false positives, increased accuracy for ambiguous samples, as well as better understanding of malware behavior during the early stages. The real-time results demonstrate the practicality of using the model in real-world environments. Overall, the proposed model provides an efficient, scalable, and explainable solution for modern malware detection.

Future Scope

The proposed hybrid ML-LLM model shows strong performance; however, it could be enhanced with several modifications to become more effective and practical. Future directions may include:

- Integration of advanced deep learning models: Employment of transformer-based sequence models or graph neural networks would probably yield better results in detecting difficult API-call patterns and long-range dependency chains.
- Expansion to large and diverse datasets: Inclusion of additional ransomware families, zero-day samples, and cloud-native malware would increase robustness and generalization capabilities.
- End-to-end automated sandboxing pipeline: This would better scale behavioral log capture in real time without requiring manual preprocessing of files.
- Fine-tuned cybersecurity LLMs: This may help improve the reasoning capabilities of domain-specific LLMs in malware behavioral reports, thus reducing hallucinations in threat interpretations.
- Real-world system integration: Integrating the model with SOC environments and EDR solutions would enable testing in enterprise environments and evaluation under real-time workloads.
- Lightweight on-device detection: Making the framework lightweight can provide protection for edge devices or IoT devices that are resource-constrained.

This set of directions clearly has tremendous potential for further developing the proposed hybrid system in a more comprehensive manner.

References

1. AV-Test Institute, “Malware Statistics & Trend Report,” 2024. [Online]. Available: https://www.av-test.org
2. Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel, “A Survey on Automated Dynamic Malware Analysis Techniques,” ACM Computing Surveys, vol. 44, no. 2, pp. 1–42, 2012.
3. Ulrich Bayer, Andreas Moser, Christopher Kruegel, and Engin Kirda, “Dynamic Analysis of Malicious Code,” Journal in Computer Virology, vol. 2, no. 1, pp. 67–77, 2006.
4. Xiaoyang Zhang, Li Wei, Jun Li, and Hao Chen, “BEACON: LLM-Based Malware Detection Using Sandbox Logs,” IEEE Transactions on Information Forensics and Security, 2025.
5. Youngmin Kim, Seojin Park, Minho Lee, and Jaehoon Kim, “MalAware: Large Language Models for Malware Behavior Summarization,” Proceedings of the IEEE Security and Privacy Workshops, 2025.
6. Rui Wang, Qiang Li, Zhenyu Chen, and Ming Zhao, “Temporal Graph-Based Ransomware Detection,” IEEE Access, vol. 13, pp. 11234–11245, 2025.
7. Saurabh Patel and Ankit Verma, “Large Language Models in Malware Analysis: A Survey,” IEEE Access, vol. 12, pp. 99876–99892, 2024.
8. John Smith, Alice Brown, Michael Johnson, and David Lee, “Early Detection of Ransomware Using Machine Learning,” Computers & Security, vol. 118, 2023.