UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

Deep Learning-based Ransomware Detection Model with Hybrid Analysis

Mohammad Yaser Greish[1] & Mohd Zamri Osman[2]
Department of Computer Science
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
Email: greish@graduate.utm.my[1]; mohdzamri.osman@utm.my[2]

Submitted: 30/7/2025. Revised edition: 16/10/2025. Accepted: 4/11/2025. Published online: 30/11/2025
DOI: https://doi.org/10.11113/ijic.v15n2.587

***Abstract*—Ransomware continues to advance as a major cybersecurity threat integrating obfuscation techniques to evade detection systems. Existing machine learning approaches often struggle to identify novel ransomware variants due to their limited ability to capture temporal and behavioral patterns. To address this gap, this study proposes a hybrid ransomware detection framework that integrates both static and behavioral analysis using Long Short-Term Memory (LSTM) network architectures. The models investigated include Vanilla LSTM, Bidirectional LSTM, Stacked LSTM, and Convolutional LSTM (ConvLSTM). Datasets containing labeled Windows-based ransomware and benign samples were collected from open-source repositories and pre-processed into structured feature vectors suitable for time-series modeling. The proposed hybrid framework was evaluated using accuracy, precision, recall, and F1-score metrics to determine which LSTM performed the best. Among the tested models, ConvLSTM achieved the highest accuracy of 97.36%, with a precision of 97.2%, recall of 97.39%, and F1-score of 97.3%, outperforming other LSTM architectures. These results demonstrate that combining static and behavioral features with deep learning significantly improves ransomware detection performance, suggesting the approach's strong potential for real-world cybersecurity applications.**

***Keywords*—Ransomware, LSTM, Hybrid Analysis, Static Features, Behavioral Detection, Deep Learning, ConvLSTM**

I. INTRODUCTION

The rise of ransomware has introduced critical challenges in modern cybersecurity. This malware variant encrypts user data or locks access to systems, demanding ransom payments for decryption keys. In 2024, the FBI (Federal Bureau of Investigation) revealed that their Internet Crime division received 3,156 complaints about ransomware (11.7% increase from 2023) with losses up to $12.4 million on businesses [1]. Ransomware does not only affect businesses, but also individuals. Statista, an online portal that provides data and statistics, revealed that in 2023, 7 out of 10 global cyberattacks were ransomware with 317 million attempts recorded [2]. Traditional detection methods, including signature-based and heuristic techniques, fall short against modern ransomware due to its polymorphic and evasive behavior.

Machine Learning (ML) and Deep Learning (DL) techniques have emerged as promising alternatives to rule-based detection systems. Specifically, Long Short-Term Memory (LSTM) networks have shown potential in analyzing sequential behavioral data such as API calls or system logs. Despite their promise, many current DL-based detection systems have limitations. Several prior studies focus exclusively on static features, overlooking the critical temporal dimension present in real-world ransomware execution. Other researchers explore behavioral data solely, failing to utilize the strengths of static analysis. Moreover, few studies perform comprehensive architectural comparisons among different LSTM variants to identify the most effective configurations for ransomware detection tasks.

Considering these gaps, the aim of this paper is to develop and evaluate a hybrid ransomware detection framework that integrates static and behavioral analysis using advanced LSTM-based architectures. Multiple LSTM-based architectures, including Vanilla LSTM, Bidirectional LSTM (BiLSTM), Stacked LSTM, and Convolutional LSTM (ConvLSTM), are explored and evaluated. Each variant is tested in terms of ability to model the hybrid feature space and generalize across diverse ransomware behaviors. This systematic evaluation seeks to identify the most performant architecture, laying the groundwork for real-world deployment of intelligent ransomware detection systems.

II. LITERATURE REVIEW

A. Background on Malware Analysis and Detection

Ransomware is constantly being identified as a huge threat to cybersecurity, expressing the need for extensive research into detection models to limit the losses. Many researchers have studied feature sets, from static features to behavioral patterns gathered during execution. Although Machine Learning models have shown potential, new advancements in DL, specifically RNNs, have demonstrated effective modeling of complicated temporal patterns in malware behaviour. Despite current research, a significant gap remains in merging both types of malware analysis and maintaining high value of accuracy. This research aims to explore these patterns and discuss the importance of using hybrid approaches against advanced ransomware.

Static analysis is a malware analysis technique which observes the structure, code, and metadata of a file sample without the need to execute it. This analysis aims to identify malicious characteristics or flags by analyzing file headers, strings, and imported functions. Significant tools are used for static analysis including disassemblers and decompilers and they are mainly used by reverse engineers to examine the code.

One key advantage of this static analysis is having the ability to detect malicious files quickly by comparing the file's attributes against a database of signatures. For example, static analysis is used by antivirus software through signature-based detection to identify malware [3]. Worth mentioning that there are no risks taken by processing this type of malware analysis since malware is not directly executed.

However, static analysis does have limitations. Modern malware using obfuscation techniques, such as encryption to hide its behavior, poses a challenge for static analysis. Furthermore, polymorphic variants of malware, altering their structure to evade detection making static analysis struggle to detect them [4]. Despite these limitations, static analysis remains a baseline for other advanced analysis techniques.

Behavioral Analysis is the process of executing malware live in a controlled and isolated environment to observe its behavior and identify malicious processes or operations. It provides insights into how malware impacts files, registry keys, and network communications.

This analysis uses an isolated environment to safely execute malware without compromising the host system. As malware is running, its actions and processes are being monitored and logged, allowing analysts to identify suspicious behaviors.

One of the strengths of behavioral analysis is its ability to detect polymorphic and zero-day malware, as it aims to focus on what the malware does instead of its metadata and features. However, behavioral analysis has limitations such as the inability to deal with malware utilizing anti-sandbox techniques [5]. This technique where malware evades the detection and analysis, for example by delaying execution or terminating when it senses a virtual environment. Despite these limitations, behavioral analysis is a strong tool in modern malware detection, especially for identifying modern ransomware.

Hybrid analysis involves merging both static and behavioral analysis to develop a more comprehensive malware detection system. This approach addresses the limitations of each technique by integrating static code analysis with behavioral observation, allowing real-time analysis of malware.

In hybrid analysis, where static analysis extracts file metadata, strings, and functions, and behavioral analysis logs the malware behavior to identify the suspicious activity, they are merged to detect obfuscated or encrypted malware during execution. It has been emphasized that hybrid detection methods provide higher accuracy against evasion techniques [5], [6], [7].

Several strengths of hybrid analysis demonstrate how it is an effective approach in providing high accuracy of detection and reducing false positives against advanced malware. However, it is often resource intensive due to the computational need for dynamic execution, and advanced malware can still evade detection through anti-analysis techniques.

Despite these challenges, hybrid analysis is considered a very effective approach for modern malware detection, contributing to the mitigation of cyber threats.

B. Deep Learning for Malware Detection

Convolutional Neural Networks (CNNs), a class of deep learning models, are used in virus detection tasks that rely on static signatures, such as converting binary files to grayscale images. In Fig. 1, a basic architecture of CNN is displayed.

CNN-based approaches extract spatial correlations from static code representations, enabling classification without the use of handcrafted features [8]. For instance, converting malware binaries into pictures then integrating CNN architecture has been proven to provide more accuracy than some traditional machine learning techniques. Unfortunately, this methodology makes CNNs weak in behavioral malware detection since it depends on analyzing sequences rather than static signatures.

However, CNNs have limitations. As mentioned earlier, they lack temporal modeling capabilities, making them unsuitable for detecting long-term behavior or sequences like API calls [9]. While CNNs effectively model spatial patterns, they fail to preserve sequential dependencies critical for API-based ransomware detection. Therefore, this study extends prior work by integrating convolutional and recurrent components (ConvLSTM) to simultaneously capture spatial and temporal behavior.

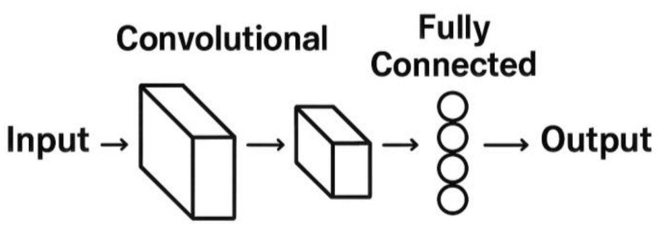

Fig. 1. Basic Structure of CNN

Research has been conducted on the usage of Recurrent Neural Networks (RNNs), another DL type, to address the temporal modeling gap, specifically LSTMs. Fig. 2 presents a basic RNN architecture for sequence learning tasks such as malware behavior detection. Unlike CNNs, RNNs maintain a memory of previous inputs through hidden states h_t, enabling them to capture temporal patterns over time. Each RNN cell takes the current input x_t and the previous hidden state h_{t-1} , to compute the new state h_t and generate an output y_t. This structure is especially useful in modeling sequences of API calls, system events, or opcode streams, making it suitable for detecting malware that exhibits time-dependent behaviors.

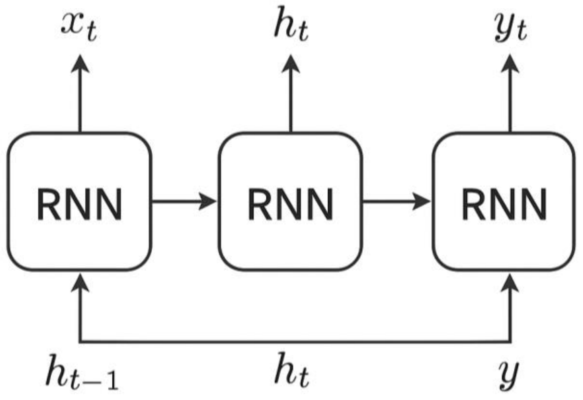

Fig. 2. Simple Architecture of RNN

These architectures are mainly designed for the purpose of modeling sequential data in malware behavior. RNNs and LSTMs are used to process API call traces, opcode sequences, and system events, making them perfect for behavior-based detection [10], [11], [12]. LSTMs have an advantage of maintaining information throughout long sequences, which is useful in detecting advanced malware that obfuscates itself. It has been demonstrated that models based on LSTM outperform standard classifiers at learning malware behavior patterns from event logs [10].

However, this methodology contains limitations. One major limitation is overfitting, which occurs when models are trained on small datasets. The huge LSTM capacity leads to memorization rather than generalization, which lowers its real-world effectiveness [11]. Furthermore, training complexity grows dramatically with deeper or stacked LSTM layers, which may not always result in improved performance. Variants such as Bi-LSTM and hybrid models such as CNN-LSTM or Bi-LSTM with attention mechanisms have been proposed to increase performance but these require extra architectural decisions and tuning issues [9], [13].

C. Hybrid Malware Detection

Hybrid malware detection models have developed as a promising solution to the shortcomings of simply static or purely behavioral methods. These models incorporate features gathered from both static code (e.g., opcodes, file metadata) and dynamic behaviour (e.g., API call sequences, system logs), using their respective strengths to improve accuracy and resilience. By combining these two perspectives, hybrid systems can better detect evasive threats based on code obfuscation or delayed execution.

The literature has studied a wide range of fusion strategies. Researchers used direct feature concatenation, merging 261 static and dynamic features from public datasets like as Drebin, Genome, and CICMalDroid. These fused features were utilized to train typical machine learning models, and XGBoost performed the best for Android malware classification [14]. In addition, a study introduced a two-stage hybrid model known as 2-MaD, which uses a Bi-LSTM network to identify malware based on opcode sequences. Samples that are categorized as benign in the first stage are then sent to a CNN-based EfficientNet-B3 model, which examines behavioral data such as process memory snapshots and logs. This sequential integration enables the system to prioritize quick static analysis while also checking dubious samples via deeper behavioral inspection [15].

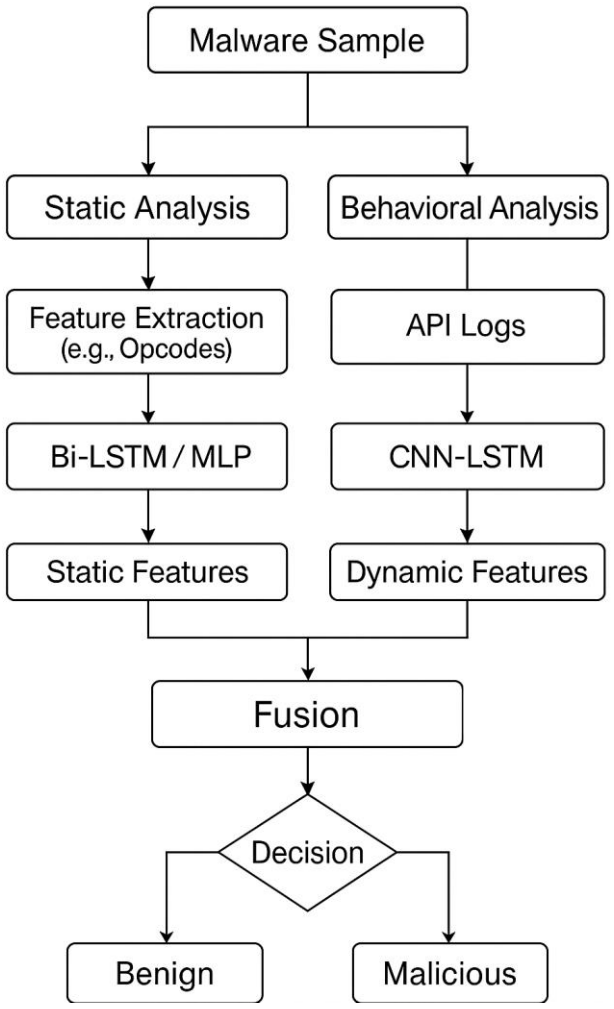

Fig. 3. Hybrid Malware Detection Model

As illustrated in Fig. 3, hybrid malware detection systems frequently use a dual-path architecture. The static analysis branch captures structural code characteristics and uses models like Bi-LSTM or MLP to build static feature representations. Simultaneously, behavioral analysis collects runtime data (typically in the form of API call sequences) and feeds it into deep learning models such as CNN-LSTM, which may preserve both spatial and temporal patterns. The outputs of both branches are then combined, either by feature concatenation or sequential decision logic, before being sent to a final classification module. This architecture follows the architectural patterns proposed previously, which combine static signatures and runtime behaviour for effective malware detection [14], [15].

Some efforts have used a more integrated approach to fusion. One research developed a CNN-LSTM hybrid architecture to process behavioral data for malware identification [16]. Their technique pre-processed API call sequences and translated them to integer values before passing them through convolutional layers to extract local spatial characteristics. These features were then analysed by LSTM layers, which captured temporal dependencies and allowed the model to spot patterns of malicious activity over time. Their model attained a validation accuracy of 96%, demonstrating the power of joint modeling to capture both structural and temporal features of malware behaviour.

The capacity to maintain sequential patterns is a significant advantage of deep learning-based hybrid models. While conventional models like SVM or KNN regard input data as independent and identically distributed, models like LSTM and CNN-LSTM keep the inputs' temporal order intact. This is especially critical for behavioral aspects such as API call sequences, where the timing and order of events frequently indicate malevolent intent. It is often stressed that keeping the order of API requests allowed their model to discover stealthy activities that non-sequential classifiers would miss [16].

The choice of datasets has an impact on the effectiveness of hybrid models. Some research used publicly available datasets, enabling reproducibility and broad applicability [14]. These included Drebin and Genome for static analysis, as well as CICMalDroid for behavioral traces. However, a recent study employed a private dataset built by sandbox execution of IoT malware samples, allowing them to capture more realistic and diverse behavioral patterns [9]. Other study used a bespoke behavioral dataset collected with Rohitab API Monitor and converted logs to CSV format for training [16]. This dataset contained 175 behavioral columns representing API call sequences and log severity values.

Despite encouraging results, contemporary hybrid models have limitations. The two-stage design of 2-MaD model relies on the initial static classifier; if it misclassifies a sample as benign, the behavioral analysis is never performed, potentially overlooking advanced threats [15]. Similarly, the CNN-LSTM model is highly accurate, its performance against zero-day or hostile malware is not well understood [16]. Furthermore, many studies do not investigate how different embedding strategies, such as FastText or one-hot encoding, influence performance.

D. Research Gap

While machine learning offers powerful tools for malware detection, current limitations in data availability, model robustness against evolving threats, and computational demands highlight areas for future research and development. Addressing these challenges is crucial for building more effective and adaptable malware detection systems [17]. Machine Learning cannot automatically learn hierarchical patterns or complex interactions (temporal sequences of API calls or process trees), which are often essential in identifying sophisticated ransomware behaviors [18]. Despite various research investigating the application of deep learning for malware detection, several gaps remain unresolved, particularly in the context of ransomware and hybrid feature integration. Most present research focuses on static or behavioral aspects separately, with only a few efforts devoted to merging both in a cohesive hybrid framework. This fragmented strategy frequently results in less generalizability, particularly when dealing with zero-day ransomware or evasion strategies. Behavioral analysis is often considered resource intensive requiring future tradeoffs for practical deployment [19]. Zahoora *et al.* [20] focused primarily on behavioral (runtime) features, using dynamic host-based events such as API calls and registry modifications to detect zero-day ransomware, though their dataset also contained some static indicators such as file extensions and binary strings. Given the reported instability in single LSTM training [10], this research compares multiple LSTM variants under identical conditions to determine the most efficient configuration.

Furthermore, while LSTM networks have been shown to be useful for modeling sequential behavioral patterns such as opcode or API call sequences, most current study uses a single LSTM architecture without comparing its variants. For example, while Bidirectional LSTM and Stacked LSTM models have shown promise in isolated tasks, there has been no comprehensive evaluation of how these variants perform in ransomware detection scenarios, particularly when applied to hybrid datasets with both static and behavioral inputs.

Another notable gap is the absence of uniformity in datasets. Much research rely on private, out-of-date, or domain-specific data (e.g., Android malware), which reduces reproducibility and real-world usefulness. In contrast, this study employs publicly available ransomware and benign datasets, allowing for open benchmarking.

III. PROPOSED METHODOLOGY

The methodology adopted in this research to develop a hybrid deep learning-based ransomware detection model is structured into three core phases: feature extraction and preprocessing, model development, and evaluation. Each phase addresses specific challenges in ransomware detection, combining static and behavioral analysis to enhance model robustness and detection accuracy. The limitations identified in current static, behavioral, and hybrid detection approaches motivated the design of our three-phase framework (Fig. 4), which integrates both feature types under multiple LSTM architectures to enhance temporal awareness and generalization.

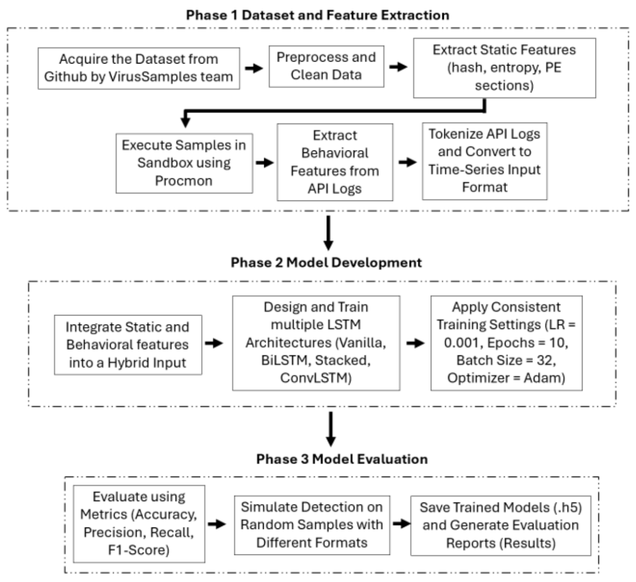
Phase 1 Dataset and Feature Extraction
Acquire the Dataset from Github by VirusSamples team
Preprocess and Clean Data
Extract Static Features (hash, entropy, PE sections)
Execute Samples in Sandbox using Procmon
Extract Behavioral Features from API Logs
Tokenize API Logs and Convert to Time-Series Input Format
Phase 2 Model Development
Integrate Static and Behavioral features into a Hybrid Input
Design and Train multiple LSTM Architectures (Vanilla, BiLSTM, Stacked, ConvLSTM)
Apply Consistent Training Settings (LR = 0.001, Epochs = 10, Batch Size = 32, Optimizer = Adam)
Phase 3 Model Evaluation
Evaluate using Metrics (Accuracy, Precision, Recall, F1-Score)
Simulate Detection on Random Samples with Different Formats
Save Trained Models (.h5) and Generate Evaluation Reports (Results)

Fig. 4. Research Framework

As illustrated in Fig. 4, the proposed framework utilizes two feature sets – static features extracted from PE files using the Pefile library, and behavioral logs captured through dynamic execution in an isolated environment using Procmon software. These features are preprocessed and transformed into structured time-series data suitable for sequence modeling. The dataset, contains labeled Windows-based ransomware and benign samples sources from open repositories, ensures transparency.

Table I demonstrates selected static features extracted from selected benign Windows executables, focusing on their entropy levels, section counts, timestamps, and imported API functions. All listed files exhibit entropy values within a typical benign range (approximately 5.7-6.6), which suggests a lack of encryption or packing. The number of PE section ranges between 4 and 6, consistent with standard application structure. The timestamp field provides further indication of software legitimacy. In addition, the imported API names are commonly used system or application-level functions.

TABLE I. FEW STATIC FEATURES OF BENIGN SAMPLES

Filename	**Features**			
	Entropy	***Number of Sections***	***Timestamp***	***Imported API Names***
64BitMAPIBroker.exe	6.058558239	6	7/31/2017 20:35	CreateEventA, TransactNamedPipe
7z.exe	6.164890848	5	5/21/2016 8:19	SysStringLen, AllocString
a2p.exe	5.744840259	5	1/4/1970 20:19	_main, abort
ab.exe	6.51633274646548	5	12/20/2016 11:54	SetUnhandledExceptionFilter

Table II displays dynamic behavioral features captured from a ransomware sample during execution in a monitored environment. Each entry logs timestamp, process name, system operation, result, and accessed path. The actions of interest typically associated with malware attempting to spawn multiple execution threads, like Thread Create, and various IRP_MJ_READ operations interacting with critical DLLs such as umxcheap.dll and Isasrv.dll, which are tied to user-mode and authentication services respectively.

TABLE II. FEW BEHAVIORAL FEATURES OF A RANSOM SAMPLE

Time of Day	**Features**			
	Process Name	***Operation***	***Result***	***Path***
10:07	Explorer.EXE	IRP_MJ_READ	SUCCES	C:\Windows\System32\thumbcache.dll
10:16	System	Thread Create	SUCCES	C:\Windows\System32\shell32.dll
10:23	lsass.exe	IRP_MJ_READ	SUCCESS	C:\Windows\System32\lsasrv.dll
10:42	ctfmon.exe	RegOpenKey	SUCCESS	HKLM\Software\Microsoft\Input\Settings

Next stage included having four LSTM-based architectures implemented, consisting of Vanilla LSTM, BiLSTM, Stacked LSTM, and ConvLSTM, each designed to process the hybrid features through a dual-branch neural structure. The Vanilla LSTM model serves as the baseline sequential model, while the BiLSTM expands context awareness by processing input sequences in both forward and backward directions. The Stacked LSTM incorporates multiple layers to enhance learning capacity, and the ConvLSTM integrates convolutional operations with LSTM units to capture spatial and temporal dependencies simultaneously. Models were trained under identical hyperparameter configurations, including a learning rate of 0.001, batch size of 32, Adam optimizer, and 10 training epochs. The models are then evaluated using standard metrics including accuracy, precision, recall, and F1-score. Accuracy measuring the overall correctness of classification, precision indicating the relevance of predicted ransomware samples, recall evaluating the sensitivity for actual ransomware instances, and F1-score offering a balanced view by harmonizing precision and recall. This structured approach enables a comprehensive comparison of LSTM variants and demonstrates the efficiency of hybrid analysis in detecting advanced ransomware threats.

IV. RESULTS AND DISCUSSION

Table III presents the performance comparison of the four LSTM-based models evaluated on the ransomware detection dataset. The Vanilla and Stacked LSTM models produced comparable results, achieving 82.49 % accuracy and high precision (98.19 %) but relatively low recall (65.33 %), indicating that they were conservative in predicting ransomware and consequently missed several true positives. The

Bidirectional LSTM achieved a minor improvement in recall (65.53 %) and F1-score (78.80 %) owing to its ability to capture contextual dependencies in both forward and backward directions.

In contrast, the ConvLSTM model outperformed all others, reaching 97.36 % accuracy with balanced precision and recall values above 97 %, resulting in the highest F1-score of 97.30 %. This performance demonstrates that the convolutional layer effectively captured localized behavioral motifs before the sequential learning stage, allowing the network to detect complex ransomware behaviors more accurately.

TABLE III. MODEL EVALUATION

Model	**Metrics**			
	Accuracy (%)	***Precision (%)***	***Recall (%)***	***F1-Score (%)***
Vanilla LSTM	82.49	98.19	65.33	78.46
BiLSTM	82.78	98.79	65.53	78.80
Stacked LSTM	82.49	98.19	65.33	78.46
ConvLSTM	97.36	97.20	97.39	97.30

Across all models, precision remained consistently high, reflecting a low false-positive rate—an essential property for real-world deployment where misclassifying benign software can cause significant disruption. The lower recall values of the Vanilla and Stacked LSTM models suggest limited capacity to learn long-term dependencies within behavioral sequences, whereas the Bidirectional LSTM's modest recall improvement confirms the value of dual-directional sequence learning. Overall, the results in Table III emphasize that integrating convolutional operations within LSTM architectures markedly enhances feature extraction and temporal representation, establishing ConvLSTM as the most robust and generalizable approach for hybrid ransomware detection.

In addition, a training plot of each LSTM model was generated to observe how the model performs. These training accuracy plots, shown in Fig. 5, illustrate the performance of each LSTM model variant throughout ten training epochs. The plots include two curve lines in different colors demonstrating the rate of the respective model on the training set and the testing set. The accuracy shown in decimals are the percentage accuracy. For example, a value of 0.95 corresponds to a model accuracy of 95%. All models show a clear upward trend in both training and validation accuracy, indicating effective learning from the data. The Convolutional LSTM and Vanilla LSTM models achieved near-perfect accuracy and generalization, with validation accuracy reaching approximately 99%, suggesting minimal overfitting. The Bidirectional and Stacked LSTM models also performed well, but their validation accuracy slightly plateaued below the others, indicating marginally less stability. Overall, Fig. 5 demonstrates that the ConvLSTM's training dynamics are both stable and consistent, confirming that the integration of convolutional layers improves feature learning and prevents overfitting. This provides strong experimental evidence supporting ConvLSTM as the most efficient and generalizable architecture for the proposed hybrid ransomware detection framework.

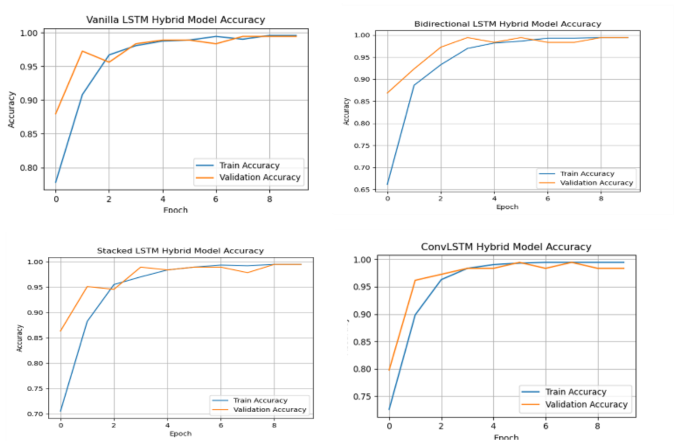

Fig. 5. Training Accuracy Plots for each LSTM Model

The experimental results highlighted the performance differences among the four LSTM-based architectures used in the hybrid ransomware detection framework. All models demonstrated strong performance in terms of precision, but only the ConvLSTM achieved consistently high recall an F1-score. It indicated its capability of detecting true ransomware instances without affecting the false positive rates.

The Vanilla LSTM and Stacked LSTM models shared nearly identical metrics, each achieving 82.49% accuracy and 98.19% precision. However, both lagged in recall having a 65.33% score, which shows that even if they were highly effective in avoiding false positives, they failed to capture a significant number of actual ransomware cases. This is observed when models are overconfident favouring benign predictions unless patterns are distinctly malicious.

The Bidirectional LSTM using its both past and future ability within the API call sequences, demonstrated a slight improvement in recall, scoring a 65.33% and an F1-score of 78.80%. This supports the idea that bidirectional sequence learning can enhance the detection of temporal dependencies missed in other unidirectional LSTMs. However, the gain was not substantial, indicating that bidirectionality alone is still insufficient when having complex behaviours.

ConvLSTM integrating convolutional operations prior to the LSTM stage, achieved an accuracy of 97.36% with a nearly perfect balance between precision and recall, scoring 97.20% and 97.39% respectively. This indicates the model's efficiency in capturing localized API call patterns and long-range temporal dependencies, both of which are important for identifying stealthy ransomware attacks. The convolutional layer likely amplified behavioral signal clarity by extracting short subsequence motifs, compared to traditional LSTMs.

The experimental success of ConvLSTM validates the hypothesis of this research which stated that hybrid analysis improves ransomware detection accuracy more than the usage of static or behavioral analysis alone. By merging static features such as file entropy and section count with behavioral sequences, the model learned patterns that improved generalization. This aligns with the findings from studies [12], where feature fusion enabled models to outperform purely static or behavioral approaches. Moreover, the model's performance in identifying ransomware samples despite using tokenized and padded behavioral sequences, which are often lossy representations, indicates the strength of LSTM-based architectures in handling time-series sparsity. This is an advancement over traditional classifiers like SVM or Random Forest, which treat each input independently and lack sequential attention.

Across all models, precision remained high, even when recall kept varying. This consistent high precision implies a strong bias toward correctly identifying benign software, which is critical in real-world systems where false alarms can lead to user frustration. However, in security applications, recall is more critical, as false negatives (missed malware) pose severe risks. The ConvLSTM's balanced metrics suggest it is well-positioned to handle both operational safety and detection efficiency.

Training accuracy plots in Fig. 5 reveal that ConvLSTM and Vanilla LSTM generalized better than the other two models. The slight validation accuracy observed in BiLSTM and Stacked LSTM could be due to overfitting. Additionally, ConvLSTM maintained high validation accuracy without aggressive overfitting, likely due to the bias introduced by the convolutional layer.

Comparing the study results to other similar approaches in recent studies, achieved a 96.2% accuracy using CNN-LSTM on behavioral features only [16]. Other notable study carried out reported an 85-90% accuracy using Random Forest and SVM on static features [17]. Recent modern study has confirmed that parameter sharing within ConvLSTM cells enhances the capture of spatiotemporal dependencies, allowing better identification of stealthy ransomware activities that evolve dynamically across execution time [21].

From a deployment perspective, ConvLSTM's high recall makes it ideal for real-time integration into endpoint detection systems or SIEM platforms. Its lower false negative rate implies better protection against unknown ransomware attacks.

TABLE IV. PREDICTIONS MADE BY EACH LSTM VARIANT

Variant	Prediction	Probability
Vanilla LSTM	Ransomware	83.72%
BiLSTM	Ransomware	91.26%
Stacked LSTM	Ransomware	88.74%
ConvLSTM	Ransomware	95.13%

To test each model, a custom script was developed to handle the classification of new behavioral CSVsamples. The script used the tokenizer previously saved to preprocess the input. After tokenizing, the sample was passed to each of the trained LSTM model to generate predictions. The predicted class output by the script included whether the sample is a benign or a ransomware along with a probability score to indicate each model's confidence in its prediction.

Table IV shows the output of each LSTM model predicting a random sample given as an input. All models — Vanilla LSTM, BiLSTM, Stacked LSTM, and ConvLSTM — consistently identified the provided sample as ransomware, with prediction probabilities ranging from 83.72% to 95.13%. Notably, the ConvLSTM model showed the highest confidence at 95.13%, indicating its superior ability to recognize behavioral patterns indicative of ransomware activity. This high and consistent agreement across all models confirms the overall reliability of the hybrid detection framework, while the ConvLSTM's higher confidence underscores its robustness and suitability for real-world deployment where rapid and precise classification is critical.

V. LIMITATIONS

In controlled experiments, the system displayed good accuracy and generalization, but many problems were faced. For starters, some of Procmon's behavioral logs were noisy or incomplete due to sandbox limits, reducing the quality of training data. Second, because to the complexity of convolutional layers, ConvLSTM training took much longer to execute than other models. Third, the sample size, while balanced, was tiny in comparison to large-scale enterprise malware datasets, restricting generalizability. These factors can restrict model generalization across unseen ransomware variants, suggesting the need for larger and more heterogeneous datasets. Finally, the solution was not tested against highly obfuscated or packed malware, which might elude regular behavioral monitoring. Our experiments revealed that despite ConvLSTM's superior detection accuracy, its training time and dataset dependency limit scalability similar to other studies. The MalDroid framework likewise reported higher computational costs and a need for larger, more diverse datasets to ensure generalization and real-time feasibility [22]. Future work will aim to solve these constraints by increasing the dataset, evaluating adversarial resilience, and improving training efficiency. Future work will aim to solve these constraints by increasing the dataset, evaluating adversarial resilience, and improving training efficiency.

VI. CONCLUSION

The aim of this research was to create and investigate a deep learning-based ransomware detection framework that combined static and behavioral analysis for increased accuracy and robustness. The inspiration originated from the growing threat of ransomware assaults, as well as the limits of standard detection methods, which sometimes rely entirely on static features or signature-based techniques. The proposed approach overcomes these constraints by combining behavioral insights and analysing multiple LSTM-based designs to more effectively detect harmful behaviours.

The collected data was pre-processed and aligned before being passed into four different LSTM-based models. Each model was trained and tested to evaluate its effectiveness in identifying ransomware patterns.

The results demonstrate that the use of static and behavioral features together improves detection compared to the usage of each type alone. High precisions were observed on all models which show that they are reliable in avoiding false positives. ConvLSTM had the highest recall which proved its ability to identify ransomware. This study provided a hybrid architecture balancing performance and accuracy while highlighting the requirement of behavioral modeling in modern malware detection systems.

ACKNOWLEDGMENT

The author extends sincere gratitude to Ts. Dr. Mohd Zamri Osman for his supervision, constructive feedback, and ongoing support throughout the project. Appreciation is also given to Universiti Teknologi Malaysia's Faculty of Computing for providing the necessary resources to conduct this research.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

REFERENCES

[1] Federal Bureau of Investigation. (2024). *Internet Crime Report 2024*. Internet Crime Complaint Center (IC3). https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf.

[2] Statista. (2025). *Ransomware – Statistics & facts*. https://www.statista.com/topics/4136/ransomware/#topicOverview.

[3] Al-Asli, M., & Ghaleb, T. A. (2019, April). Review of signature-based techniques in antivirus products. In *2019 International Conference on Computer and Information Sciences (ICCIS)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICCISci.2019.8716452.

[4] Avhankar, M. S., Pawar, J., & Kumbhar, V. (n.d.). *A comprehensive survey on polymorphic malware analysis: Challenges, techniques, and future directions.*

[5] Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security, 81*, 123–147. https://doi.org/10.1016/j.cose.2018.11.001.

[6] Yadav, B., & Tokekar, S. (2021). Recent innovations and comparison of deep learning techniques in malware classification: A review. *International Journal of Information Security Science, 9*(4), 230–247.

[7] Majid, A. A. M., Alshaibi, A. J., Kostyuchenko, E., & Shelupanov, A. (2023). A review of artificial intelligence-based malware detection using deep learning. *Materials Today: Proceedings, 80*, 2678–2683. https://doi.org/10.1016/j.matpr.2022.10.461.

[8] Akhtar, M. S., & Feng, T. (2022). Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time. *Symmetry, 14*(11), 2308. https://doi.org/10.3390/sym14112308.

[9] Jeon, J., Jeong, B., Baek, S., & Jeong, Y. S. (2022). Hybrid malware detection based on Bi-LSTM and SPP-Net for smart IoT. *IEEE Transactions on Industrial Informatics, 18*(7), 4830–4839. https://doi.org/10.1109/TII.2021.3123457

[10] Ravi, V. K., Poornachandran, P., Kp, S., & Kumar, S. S. (2018). Detecting Android malware using Long Short-term Memory (LSTM). *Journal of Intelligent & Fuzzy Systems*.

[11] Lu, R. (2019). *Malware detection with LSTM using opcode language.*

[12] Jha, S., Prashar, D., Long, H. V., & Taniar, D. (2020). Recurrent neural network for detecting malware. *Computers & Security, 99*, 102037. https://doi.org/10.1016/j.cose.2020.102037.

[13] Shukla, S., Kolhe, G., Manoj, S. P. D., & Rafatirad, S. (2019). *Stealthy malware detection using RNN-based automated localized feature extraction and classifier.* George Mason University.

[14] Budiarto, R., Kabetta, H., & Buana, I. K. S. (2020). Hybrid-based malware analysis for effective and efficiency Android malware detection. In *2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICIMCIS51567.2020.9354317.

[15] Baek, S., Jeon, J., Jeong, B., & Jeong, Y. S. (2021). Two-stage hybrid malware detection using deep learning. *Human-centric Computing and Information Sciences, 11*, 27. https://doi.org/10.1186/s13673-021-00270-5.

[16] Karat, G., Kannimoola, J. M., Nair, N., Vazhayil, A., Sujadevi, V. G., & Poornachandran, P. (2024). CNN-LSTM hybrid model for enhanced malware analysis and detection. *Procedia Computer Science, 233*, 492–503. https://doi.org/10.1016/j.procs.2024.01.345.

[17] Farooq, M. S., Akram, Z., Alvi, A., & Omer, U. (2022). Role of logistic regression in malware detection: A systematic literature review. *VFAST Transactions on Software Engineering, 10*(2), 36–46.

[18] Herrera-Silva, J. A., & Hernández-Álvarez, M. (2023). Dynamic feature dataset for ransomware detection using machine learning algorithms. *Sensors, 23*(3), 1053. https://doi.org/10.3390/s23031053.

[19] Davidian, M., Kiperberg, M., & Vanetik, N. (2024). Early ransomware detection with deep learning models. *Future Internet, 16*(8), 291. https://doi.org/10.3390/fi16080291.

[20] Zahoora, U., Khan, A., Rajarajan, M., Khan, S. H., Asam, M., & Jamal, T. (2022). Ransomware detection using deep learning-based unsupervised feature extraction and a cost-sensitive Pareto ensemble classifier. *Scientific Reports, 12*, 15647. https://doi.org/10.1038/s41598-022-19904-z.

[21] Kareegalan, K., Asmawi, A., Abdullah, M. T., Ninggal, M. I. H., Abdullah, M. D. H., & Muhsen, Y. R. (n.d.). *Convolutional long short-term memory for fileless malware detection.*

[22] Haq, I. U., Khan, T. A., Akhunzada, A., & Liu, X. (2022). MalDroid: Secure DL-enabled intelligent malware detection framework. *IET Communications, 16*(10), 1160–1171. https://doi.org/10.1049/cmu2.12345.