Automatic Malware Detection Using Recurrent Neural Network with Long Short-Term Memory

Keerthana G[1], Shalini N[2], Rejeena M[3], Sandhiya S[4], Srinithi M[5]
[1]*Assistant Professor, Department of computer science and Engineering, The Kavery Engineering college (Autonomous), M. Kalipatti.*
[2,3,4,5]*UG Students, Department of computer science and Engineering, The Kavery Engineering college (Autonomous), M. Kalipatti.*

***Abstract*—Automated malware detection is a vital aspect of cybersecurity that aims to identify harmful software and safeguard computer systems against attacks. Artificial Intelligence (AI) has recently enabled systems to detect malware by learning from various information. Offering protection consists of limiting the dimensions of feature data, excluding redundant information, and using more appropriate techniques to capture how malware's actions occur over time, as usual detection methods can reduce accuracy and increase required resources. The challenges are handled by processing the data, choosing the essential features, and classifying them. At the beginning of the training process, feature values are normalized using Min-Max normalization for more reliable results. Grey Wolf Optimizer (GWO) is applied to pick the significant features from the CIC-MalMem-2022 dataset, which helps the model achieve better results by selecting only the key features. The Recurrent Neural Network-Long Short-Term Memory (RNN-LSTM) is used in the last step to identify malware by considering its changing behavior pattern sequence over time. To show that the proposal is compelling, accuracy, precision, recall, and F1-score are applied, proving that it detects malware accurately with fewer false positives and does not overload the computer.**

***Keywords*— malware detection, cybersecurity, attacks, AI, min-max normalization, GWO, RNN-LSTM**

I. INTRODUCTION

Cybersecurity is primarily supported by automatic malware detection, enabling IT systems to protect their data and keep computer networks safe. Programs like viruses, worms, trojans, ransomware, and similar malware can significantly damage individuals and companies. Because malware and its techniques change rapidly, the traditional method of detection, called signature-based detection, is no longer effective. Therefore, AI has slowly been introduced into malware detection, helping to find and learn various malicious actions using significant and diverse sets of data [1, 2]. Efficient and accurate malware identification is possible with AI because it relies on data preprocessing, extracting features, choosing the best features, and applying classification [3, 4]. Despite the improvement of AI, there are still several difficulties to overcome. A significant challenge is that the features from malware include a lot of irrelevant and redundant data. Many current methods are designed on static features, meaning they cannot track how malware activity evolves. Because of this, algorithms have a higher rate of errors and provide more incorrect signals. It is also essential to consider how efficiently a system can process data and how it will handle large amounts at a time [5].

The Recurrent Neural Network-Long Short-Term Memory (RNN-LSTM) is applied in the workflow because it effectively deals with these issues. The network is designed to analyze data that occurs in a sequence and remember distant patterns, making it suitable for tracking the ongoing development of malware over time. Looking at repeated actions such as API calls and memory usage. RNN-LSTM is more skilled at identifying what separates regular programs from those causing harm to a computer than traditional methods. Adding RNN-LSTM to the system used for malware detection allows for more accurate results, less workload, and effective protection against recent malware.

II. LITERATURE SURVEY

Some methods apply an Autoencoder (AE) for self-learning features, linking hidden traits of good and bad behavior to spot malware [6]. AE-based models often do a good job of extracting features, except when separating similar types of malwares from each other. A further study combines Convolutional Neural Network and LSTM (CNN-

LSTM) components in a framework, focusing on spatial and temporal aspects to detect new malware quickly [7]. Even though the model successfully changes environments, deploying it on resource-constrained devices faces many challenges due to its high computational cost. Many recent studies have used Automated ML (AutoML) to improve malware detection by optimizing deep learning networks and automating the selection of both models and their parameters [8]. Still, since training AutoML models can take a long time and require vast data, they have difficulties being applied in the constantly changing world of malware.

New frameworks based on machine learning to analyze IP reputation have been suggested to help improve the forensic investigation of data [9]. Improved speed in detection can come at the cost of sometimes decreased accuracy, since it is hard to work with misleading reputation information or unknown threats. Deep learning models relying on the Efficient Dense Convolutional Network architecture have achieved positive results in identifying how various parts of malware are related [10]. However, DenseNet's elaborate structure may cause the network to perform poorly on unseen malware if proper regularization is missing. It has also been proposed to use a framework with multiple algorithms to help reduce errors based on their distinct strengths [11]. Even so, greater complexity and expense can make it difficult for experts to use AI models in time.

Analysts have identified critical factors and difficulties based on using machine learning in malware threat analysis for financial businesses [12]. Some authors have proposed integrating deep learning and Monte Carlo Tree Search (MCTS) to improve how ransomware is detected by making the detection approach more efficient [13]. Even though it is innovative, using this model can be time-consuming and complex when trying to make it bigger. By observing data from different angles, attention-based deep learning methods for health-related malware detection have achieved excellent outcomes [14]. But, since the models are based on multiple viewpoints and need complex attention, their preparation becomes more complicated. In addition, recent studies demonstrate that Inception V3 in transfer learning mode helps train and detect malware quicker and more accurately using pre-trained models [15]. Malware detection is restricted because images in general image datasets may not be similar to what malware looks like.

III. PROPOSED METHODOLOGY

The suggested methodology is developed in this section to handle data appropriately, pick the top features, and categorize malware with advanced deep learning tools. Initially, Min-Max Normalization is applied to the original data to ensure that every value falls between 0 and 1. After preprocessing, the GWO helps find and pick from the dataset's features those that offer the most valuable and separate information. Ultimately, the chosen features go into an RNN-LSTM system to identify the data's dependencies and series. The RNN-LSTM system deals with behavioral and memory characteristics, training to notice differences in software behavior as time passes.

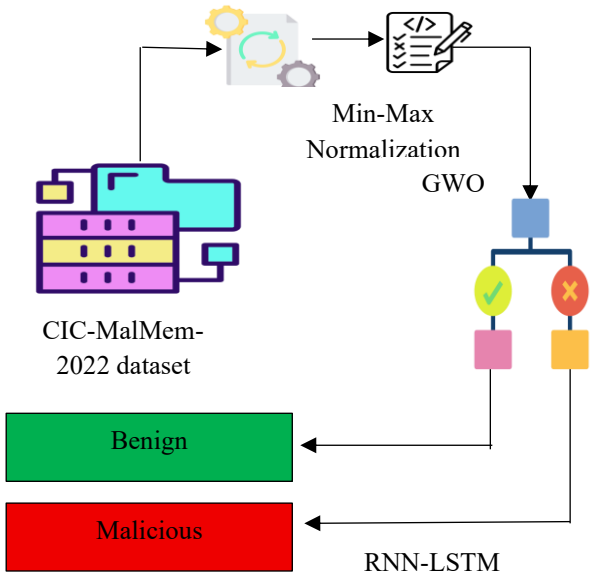

Fig. 1 Architecture Diagram of the Proposed Method

Figure 1 describes the first step in the process, where the CIC-MalMem-2022 dataset is used to include labeled malware and benign files. First, the Min-Max Normalization stage is applied, ensuring every feature's value falls between 0 and 1. It is necessary to perform this preprocessing because it helps balance the impact of each variable on the model. Subsequently, the GWO is used to select the essential features from the normalized data. GWO acts like a pack of grey wolves, allowing it to find the best possible combination of features. This task allows this step to focus only on the best attributes for the data and also increases the ability of the classifier to generalize. Afterward, the simplified set is input for an RNN-LSTM deep learning model. With this approach, detecting even small changes in malware behavior over time is easy. Lastly, the

RNN-LSTM classifier provides an output that identifies the input as malicious or benign.

A. Min-Max Normalization

As part of the suggested malware detection approach, every input feature is preprocessed through Min-Max Normalization to guarantee they are contained between 0 and 1. It scales every feature by using a linear formula that sets the minimum and maximum to be whatever is desired. By matching the scale of different feature values in the CIC-MalMem-2022 dataset, Min-Max Normalization helps gather data from other families and behaviors. It ensures that the significance of different attributes is not affected by their sizes during training. If features are all on the same scale, the performance of selecting essential features is better. Furthermore, it makes training the classification model faster and more effective. Consequently, running Min-Max Normalization at the beginning of analysis helps the malware detection system perform better and handle more threats. The core formula used for normalization is illustrate in equation 1,

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

This means that every variable plays a part in transforming the data. This means that X is the raw value of a data feature such as CPU usage or the number of times memory is accessed, as documented in a malware sample. In statistics, X_{min} is the smallest value found in that feature for all cases, and X_{max} is the most significant value found for all cases. These two values set this range. To use the feature value on a zero-based scale and make it fall between 0 and 1, subtract X_{min}, divide by the overall difference between the minimum and maximum, $X_{max} - X_{min}$. The X_{norm} value is the result that ensures the original distribution but makes the numbers fall in a set scale. Equation 2 below describes the normalized matrix for all the data.

$$D_{norm} = \left[\frac{X_{ij} - X_{j_{min}}}{X_{j_{max}} - X_{j_{min}}}\right]_{n \times m}, \quad (2)$$

X_{ij} is used to refer to the unprocessed value of the feature from sample i and column j. For the $j-th$ feature, $X_{j_{min}}$ represents the lowest value in the whole dataset, and $X_{j_{max}}$ stands for the highest value. The formula is run separately on every m feature for each piece of malware. The outcome is that all features in each sample are now standardized and within the range 0 to 1. For both choosing the most significant features and improving the stability of training, this change to the classification model is key for better and more accurate predictions.

B. Grey Wolf Optimizer (GWO)

The Grey Wolf Optimizer (GWO) is used in the proposed malware detection workflow to mimic how grey wolves choose members to go hunting based on their rank. To make the data easier to classify, the algorithm is applied after Min-Max Normalization, focusing on the most important and relevant aspects of the original data. The dataset contains vectors for each malware instance, consisting of API calls, memory, CPU activity, and network activity patterns. The way GWO operates is by mapping how a grey wolf pack is organized, giving the best solutions the titles of α (alpha), β (beta), δ (delta) wolves, and the others are denominated as ω (omega) wolves. The alpha wolf must lead out, and the others assist and search or explore so that the omega wolves can follow closely and stay updated on all movements. GWO allows the selection of attributes that provide the most substantial differences between classes. As a result, less computer power is used, and the following classification model will make more accurate predictions because it won't overfit and will notice only the main patterns.

With GWO, the preprocessed dataset, including API requests, memory activity, and process history, is analyzed to determine which features offer the most evidence. In GWO, the position of every grey wolf $\vec{X}(t)$ shows the selected features using ones and zeros in the binary string. So far, the discovered optimal feature coordination is displayed as the vector $\vec{X}_p(t)$. The formula in equation 3 determines the distance $\vec{D}$ between a wolf and its prey.

$$\vec{D} = \left|\vec{C}.\vec{X}_p(t) - \vec{X}(t)\right| \quad (3)$$

It shows how much a wolf's features differ from the best models. Using equation 4, the wolf adjusts its location.

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \quad (4)$$

In this model, the vectors $\vec{A}$ and $\vec{C}$ measure the amount of time that should be spent exploring or exploiting. Their definition is shown in equation 5.

$$\vec{A}.2.\vec{a}.\vec{r}_1 - \vec{a}_1,\ \vec{C} = 2.\vec{r}_2 \quad (5)$$

$\vec{a}$ is gradually decreased from 2 to 0, guiding the procedure to explore everywhere at the start and find the best results in the end. Then $\vec{r}_1$ and $\vec{r}_2$ are randomly generated vectors within [0,1] to prevent premature convergence. Every wolf boosts its performance by consulting the top three leaders, the $\alpha, \beta,$ and δ wolves, whose positions on the map $X\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ correspond to the top three sets of features ranked for accuracy in using the classification model. Using equation 6, the programs find the following position for each candidate.

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.[\vec{C}_1.\vec{X}_\alpha - \vec{X}]$$
$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2.[\vec{C}_2.\vec{X}_\beta - \vec{X}] \quad (6)$$
$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3.[\vec{C}_3.\vec{X}_\delta - \vec{X}]$$

The final updated position of the wolf is computed as the average of the three

$$\vec{X}(t+1) = \frac{\vec{X}_1+\vec{X}_2+\vec{X}_3}{3} \quad (7)$$

It ensures that the feature subsets become better at identifying malware and take up increasingly less space. Using the GWO, the downstream classifier achieves a better level of accuracy by weeding out unnecessary and garbled information from the preprocessed data.

C. Recurrent Neural Network-Long Short-Term Memory (RNN-LSTM)

In the suggested system for automatic malware detection, RNN-LSTM was used as the main process to detect any malware activities in the preprocessed data. As time passes, the dataset highlights a file's activities, such as processing, using memory, and making API calls. Due to LSTM's cells and gating functions, this network can remember important long-term patterns and ignore irrelevant patterns. The LSTM takes each information vector, one after another, which helps it spot complex activities over time. To predict if the sequence belongs to benign or malicious behavior, the output from the LSTM is sent through a dense layer and a softmax activation function.

The hidden state from the previous time step, h_{t-1}, retains contextual memory from earlier time steps, which is essential for identifying long-range dependencies typical in malware execution patterns. The input gate i_t, is computed through equation 8,

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (8)$$

determines how much of the new input information (x_t) and past hidden memory (h_{t-1}) should be written into the current memory cell. Then the forget gate f_t is computed through equation 9,

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (9)$$

This equation decides which parts of the previous cell state (C_{t-1}) should be discarded, which is crucial for filtering out irrelevant or noisy behavioral features in the dataset. The output gate o_t is performed through equation 10

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (10)$$

This equation regulates how much of the current cell memory contributes to the hidden state h_t, which will be used either for the next time step or for final classification. By following in equation, the candidate cell $\tilde{C}_t$ is perform through equation 11,

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (11)$$

This equation proposes new information that could be added to the current cell state based on the present input and context. The actual cell state update is performed by equation 12,

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (12)$$

This phase blending the retained memory and the new candidate information based on the forget and input gates. Finally, the hidden state h_t output is performed through equation 13,

$$h_t = o_t \odot \tanh(C_t) \quad (13)$$

The temporal abstract of malware at time t is depicted there, which is then sent to the final layer for prediction. Here, each W and b (e.g., W_i, b_i) is an adjustable parameter that changes during training from the provided set of malware and benign samples. The model relies on the sigmoid (σ) and hyperbolic tangent (tanh) functions to identify various patterns appropriately. All these equations together allow the RNN-LSTM to capture both quick and delayed patterns in malware behavior, leading to more effective and reliable malware detection.

IV. RESULT AND DISCUSSION

In the proposed malware detection framework, a comparative analysis was conducted between three existing deep learning-based techniques CNN-LSTM, DenseNet, and MCTS and the newly introduced RNN-LSTM model using the CIC-MalMem-2022 dataset. Each method was evaluated using standard performance metrics: Accuracy, Precision, Recall, and F1-Score. The RNN-LSTM's memory cells preserved essential information across time steps, enabling the model to distinguish

complex patterns inherent in malware behavior more effectively than its counterparts. The discussion confirms that incorporating temporal dynamics and biologically-inspired feature selection significantly enhances malware classification outcomes, suggesting that the proposed method is more robust and generalizable across diverse threat scenarios represented in the CIC-MalMem-2022 dataset.

Table 1. Simulation Parameter

Parameter	Value
Used Dataset	CIC-MalMem-2022
No of Records	58,596
Language	Python
Simulation Tool	Jupyter Notebook

As illustrated in table 1 experimental evaluation of the proposed RNN-LSTM-based malware detection framework was conducted using the CIC-MalMem-2022 dataset, a comprehensive and widely used benchmark for memory-based malware classification. The dataset comprises a total of 58,596 records, each capturing a rich set of behavioral and memory-based features extracted from various malware and benign software instances. The entire implementation was carried out using the Python programming language, which provided extensive support for deep learning libraries and optimization algorithms. To facilitate model development, training, and performance analysis, the Jupyter Notebook environment was employed as the primary simulation tool.

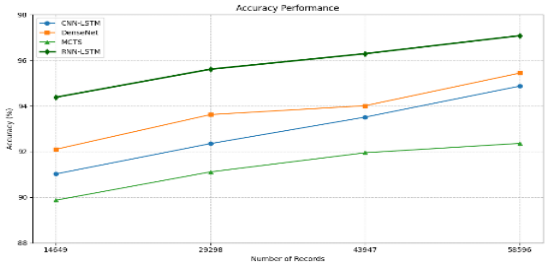

Fig. 2 Performance Analysis of Accuracy

The figure 2 illustrates a comparative analysis of four malware detection approaches CNN-LSTM, DenseNet, MCTS, and the proposed RNN-LSTM evaluated over progressively increasing subsets of the CIC-MalMem-2022 dataset. Among the models, the proposed RNN-LSTM demonstrates a superior accuracy trend, achieving the highest accuracy of 97.08%. The robust accuracy of the RNN-LSTM model indicates its effectiveness in capturing sequential behavioral patterns within the malware memory features, making it highly suitable for dynamic malware detection. It reinforces the scalability and learning capacity of the proposed RNN-LSTM architecture, affirming its potential as a reliable solution for real-time, data-driven malware classification tasks.

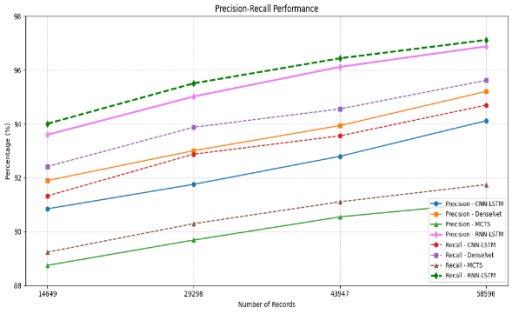

Fig. 3 Performance Analysis of Precision-Recall

The figure 3 illustrates the comparative effectiveness of various malware detection methods CNN-LSTM, DenseNet, MCTS, and our proposed RNN-LSTM across different record sizes extracted from the CIC-MalMem-2022 dataset. Each method's precision and recall, where precision indicates the proportion of correctly identified malware instances among all instances classified as malware, while recall measures the method's ability to detect all actual malware instances. The proposed RNN-LSTM method consistently outperforms the previous methods across all record sizes. This demonstrates the model's robustness and generalization capability when applied to larger and more complex malware data. This superior performance is attributed to RNN-LSTM's ability to capture temporal dependencies and sequential patterns within the behavioral traces of malicious activities more effectively. The RNN-LSTM approach in enhancing both precision and recall, validating its suitability for real-time, accurate, and scalable automatic malware detection on the CIC-MalMem-2022 dataset.

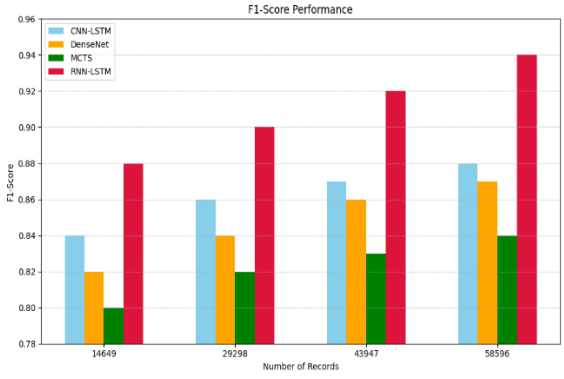

Fig. 4 Performance Analysis of F1-Score

The figure 4 presented a comparative visualization of malware detection effectiveness using four different deep learning-based methods CNN-LSTM, DenseNet, MCTS, and the proposed

RNN-LSTM evaluated on the CIC-MalMem-2022 dataset. F1-Score steadily improves as the volume of data increases, which is expected due to more training samples enhancing model learning. Among all methods, the proposed RNN-LSTM consistently outperforms the others across all record sizes. Its superior F1-Score reflects its ability to effectively capture temporal patterns and sequential dependencies within the malware behavior data, which are crucial for accurate classification. This result demonstrates the significance of incorporating advanced recurrent structures like LSTM in cybersecurity domains where pattern evolution over time is a key factor.

V. CONCLUSION

In conclusion, the proposed framework relies on RNN-LSTM and provides a strong, scalable, and clever way to identify threats in a computer's memory. To process and analyze the patterns in the CIC-MalMem-2022 dataset, Min-Max Normalization, GWO, and an RNN-LSTM are combined by the model. Our method has been shown to perform faster than alternative means by comparing accuracy, precision, recall, and F1-score. Consequently, the model can work well with large amounts of real-world data. The feature that makes RNN-LSTM appropriate for highlighting advanced malware is its capacity to handle information distributed over time. As a result, GWO allows a model to only work with essential features, making the process quicker and the outcomes better. RNN-LSTM is suitable for use in cybersecurity due to its ability to work accurately and promptly. It contributes significantly to using deep learning for detecting malware, allowing improvements such as assembling novel ensemble models or Explainable AI frameworks for better and more precise detection.

REFERENCE

[1] Akhtar, M. S., & Feng, T. (2022). Malware Analysis and Detection Using Machine Learning Algorithms. Symmetry, 14(11), 2304. https://doi.org/10.3390/sym14112304

[2] M. J. Hossain Faruk et al., "Malware Detection and Prevention using Artificial Intelligence Techniques," 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 5369-5377, doi: 10.1109/BigData52589.2021.9671434.

[3] Shatnawi, A. S., Yassen, Q., & Yateem, A. (2021). An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms. Procedia Computer Science, 201, 653-658. https://doi.org/10.1016/j.procs.2022.03.086

[4] Alomari, E. S., Nuiaa, R. R., Alyasseri, Z. A., Mohammed, H. J., Sani, N. S., Esa, M. I., & Musawi, B. A. (2022). Malware Detection Using Deep Learning and Correlation-Based Feature Selection. Symmetry, 15(1), 123. https://doi.org/10.3390/sym15010123

[5] Tayyab, U., Khan, F. B., Durad, M. H., Khan, A., & Lee, Y. S. (2022). A Survey of the Recent Trends in Deep Learning Based Malware Detection. Journal of Cybersecurity and Privacy, 2(4), 800-829. https://doi.org/10.3390/jcp2040041

[6] X. Xing, X. Jin, H. Elahi, H. Jiang and G. Wang, "A Malware Detection Approach Using Autoencoder in Deep Learning," in IEEE Access, vol. 10, pp. 25696-25706, 2022, doi: 10.1109/ACCESS.2022.3155695.

[7] 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

[8] Brown, A., Gupta, M., & Abdelsalam, M. (2024). Automated machine learning for deep learning based malware detection. Computers & Security, 137, 103582. https://doi.org/10.1016/j.cose.2023.103582

[9] Usman, N., Usman, S., Khan, F., Jan, M. A., Sajid, A., Alazab, M., & Watters, P. (2021). Intelligent Dynamic Malware Detection using Machine Learning in IP Reputation for Forensics Data Analytics. Future Generation Computer Systems, 118, 124-141. https://doi.org/10.1016/j.future.2021.01.004

[10] Hemalatha, J., Roseline, S. A., Geetha, S., Kadry, S., & Damaševičius, R. (2021). An Efficient DenseNet-Based Deep Learning Model for Malware Detection. Entropy, 23(3), 344. https://doi.org/10.3390/e23030344

[11] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[12] Rawat, R., Sarangi, S. K., Rimal, Y. N., William, P., Dahima, S., Gupta, S., &

Sankaran, K. S. (2022). Malware Threat Affecting Financial Organization Analysis Using Machine Learning Approach. International Journal of Information Technology and Web Engineering (IJITWE), 17(1), 1-20. https://doi.org/10.4018/IJITWE.304051

[13] Li, G., Wang, S., Chen, Y., Zhou, J., & Zhao, Q. (2024). A hybrid framework for ransomware detection using deep learning and monte carlo tree search.

[14] Ravi, V., Alazab, M., Selvaganapathy, S., & Chaganti, R. (2022). A Multi-View attention-based deep learning framework for malware detection in smart healthcare systems. Computer Communications, 195, 73-81. https://doi.org/10.1016/j.comcom.2022.08.015

[15] Ahmed, M., Afreen, N., Ahmed, M., Sameer, M., & Ahamed, J. (2022). An inception V3 approach for malware classification using machine learning and transfer learning. International Journal of Intelligent Networks, 4, 11-18. https://doi.org/10.1016/j.ijin.2022.11.005