Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 233 (2024) 492–503

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Innovative Data Communication Technologies and Application (ICIDCA 2024)

# CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection

Gautam Karat[a], Jinesh M. Kannimoola[b], Namrata Nair[a], Anu Vazhayil[a], Sujadevi V G[a], Prabaharan Poornachandran[a]

[a]*Centre for Internet Studies and Artificial Intelligence, Amrita Vishwa Vidyapeetham, Amritapuri, India*
[b]*Department of Computer Science and Applications, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Amritapuri, India*
gautamkarat@am.amrita.edu, jinesh@am.amrita.edu, namratan@am.amrita.edu, anuvazhayil@am.amrita.edu, sujap@am.amrita.edu, praba@am.amrita.edu

**Abstract**

The ever-evolving tactics employed by malware authors to avoid detection pose challenges to the conventional static analysis method, which entails examining the malware code. These challenges arise from the authors' capacity to obfuscate their code. To address this matter and enhance the identification of malware, integrating dynamic detection and machine learning has emerged as a highly promising approach. This methodology has demonstrated efficacy in identifying malware specifically engineered to circumvent established detection techniques. Behavioural analysis is a crucial component in ensuring endpoints' security, with the CNN-LSTM algorithm being particularly notable for its effectiveness in identifying Zero-Day malware. This type of malware poses a substantial obstacle to conventional signature-based approaches. This paper aims to assess the efficacy of the Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) model, emphasising its significance in tackling the continuously evolving realm of cybersecurity obstacles. The research highlights the significance of transitioning from traditional signature-based detection methods to behavioural analysis techniques. It suggests utilising deep learning approaches such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) to improve the ability to detect malware in an environment where threats constantly evolve. The malware detection system that has been developed encompasses a log parser analyser, API monitoring, and an extension checker module. The CNN-LSTM model demonstrates a commendable ability to accurately identify malicious behaviour, achieving a validation accuracy of 96%. This study demonstrates the efficacy of employing behavioural analysis and deep learning techniques to enhance cybersecurity, particularly in addressing sophisticated, evasive, and previously unknown malware risks.



*Keywords:* Malware Detection ; Dynamic Analysis ; CNN-LSTM ; Hybrid Model; Deep Learning


10.1016/j.procs.2024.03.239

## 1. Introduction

Within the dynamic realm of cybersecurity, the perpetual struggle against malevolent software, commonly called malware, has emerged as a persistent and enduring obstacle. Traditional antivirus systems have relied on signature-based detection methods for a long time. This means that malware is found by comparing it to patterns or signatures already recorded in large databases [1]. Although this particular method has demonstrated efficacy in the detection of established risks, it needs help keeping up with the swift growth of novel malwares and their various iterations. The increasing prevalence of advanced and elusive malware necessitates[2] the development of more resilient and flexible solutions within the realm of cybersecurity. The lack of uniformity among various antivirus providers exacerbates the flaws of conventional antivirus solutions that rely on signature-based detection. This leads to inconsistencies in the extent and currency of their signature databases [3]. The constraints, as mentioned above, give rise to issues regarding the efficacy of antivirus software in accurately identifying and mitigating malware infections. To tackle these issues, the cybersecurity community has been actively investigating new methodologies that prioritise analysing malware behaviour rather than depending exclusively on established signatures.

The examination of malware from a behavioural perspective has shown promise in bolstering cybersecurity efforts [4]. This methodology entails the meticulous observation and analysis of the activities and interrelationships exhibited by files or programmes to detect any potentially dubious or harmful conduct. The technology enables security systems to identify and address malicious actions, even when the malware's signature is unfamiliar or altered. The discipline of behavioural malware analysis has shown notable progress in enhancing the precision and effectiveness of malware detection through the incorporation of machine learning and deep learning methodologies [5, 6]. Algorithms such as Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) have demonstrated their efficacy in analysing malware by examining their behavioural patterns [7]. These methodologies are revolutionising the approach to studying malicious software and augmenting our capacity to mitigate emerging security risks. The CNN-LSTM algorithm analyses behavioural malware well. It detects polymorphic, metamorphic, and zero-day malware. Deep learning models for API call analysis can detect malware by recognising complex relationships and patterns, automating feature extraction, and efficiently processing large datasets. The research shows that these methods, along with dataset compilation and model selection, can improve malware detection and cybersecurity in a rapidly changing threat landscape.

A comprehensive system was created to detect and analyse malicious activity. All API calls during this simulation were recorded and converted into integers for CNN-LSTM model training. Log severity values were assigned. The dataset had 175 columns representing consecutive API requests for alerts and risk scores. Rohitab API Monitor facilitates the monitoring and management of API calls executed by various apps and services. Informed action categorization and severity scores were system outputs. CUDA was used for efficient training, and the CNN-LSTM model had specific layers. Thus, a script retrieved Sysmon log data from a database and generated alerts for malicious behaviour using predefined correlation rules. The logs were converted to CSV for machine learning model training. The CNN-LSTM model classified API calls from benign and malicious applications with 96% validation accuracy. Mapping API calls to integers streamlines data processing and model training. An extension checker module monitored suspicious file extensions for security. The results demonstrate the system's ability to detect and classify malicious behaviour. Visualisations of loss and accuracy during model training, an ROC curve, and a precision-recall curve prove the model's efficacy. At the same time, a confusion matrix shows high true favourable and true negative rates, proving accurate and new data classification.

## 2. Related Works

Malware detection researchers have used machine learning and deep learning to improve malware detection by analysing behavioural patterns. These studies advance cybersecurity efforts. The study by Daku et al. [8] aimed to identify new ransomware variants based on changing behaviour. Researchers classified altered ransomware iterations by analysing their behavioural patterns. Machine learning classification was used to identify ransomware samples with different family behaviours. A classification accuracy of 78% was achieved using the J48 technique, outperforming the K-Nearest Neighbour approach. Yuan et al. [9] suggest that signature-based and static analysis methods for malware detection may not be reliable. The authors implemented a system to balance the efficiency of traditional machine

learning methods, which may sacrifice accuracy, and the accuracy of deep learning approaches, which may take longer to execute. Spectrum uses conventional machine learning to monitor software behaviour and notify users when it exceeds specific criteria. It also detects malware-infected files and newly created network connections.

The researchers [10] propose using the TF-IDF measure to identify and prioritise emerging malware characteristics automatically. Behaviour logs from Cuckoo Sandbox analysis reports were used to investigate everyday and harmful actions. The authors' research helped extract ransomware components, primarily since multiple ambient logs contained ransomware activity. The researchers' method revealed the possibility of false malware indications and demonstrated their ability to detect non-malware characteristics only found in malware documents. This paper introduces a new malware detection method using convolutional neural networks (CNN) and machine learning techniques [11]. Instead of port numbers and protocols, this method uses 35 packet flow data characteristics. Malware identification became more reliable and precise after the perspective change. CNN and random forest (RF) models had 85% accuracy, precision, and recall across all malware categories. Researchers [12] aimed to identify IoT botnets during propagation. This stage involves infected devices recruiting more devices to expand the botnet. The researchers used a logistic regression model to identify IoT botnets targeting IoT devices. Data was collected from 100 botnets that expanded through brute-force attacks. The model was accurate (97.3%), precise (0.94), recall (0.98), and F-measure (0.96), proving its efficacy.

The authors in [13] explored the use of machine learning for static and dynamic malware analysis. They found that this combined method outperformed static analysis. The researchers used Cuckoo Sandbox to generate detailed output, including files, sections, and screenshots from malware execution, network activities, and antivirus findings. Among the multiple classifiers used on the dataset, the LMT classifier achieved a classification accuracy of 98.2857%. The research also found IP addresses in many files, suggesting attempts to communicate with outside parties. A comprehensive study by [14] examined machine learning-based malware detection methods. The methods were divided into static, dynamic, and hybrid. The J48 method, combined with hybrid analysis, achieved 90% accuracy in detecting malware in Windows operating systems. Combined with dynamic analysis, the Decision Tree technique achieved comparable precision in the Android operating system.The study [15] introduced behavioural malware extraction (BME), a novel approach to identifying key attributes in malware behaviour. BME outperformed unigram-based approaches, especially at different training-to-testing ratios. The study by AV (2020)[16] analysed historical antivirus software assessments to distinguish between known malware and zero-day vulnerabilities. The research findings helped develop effective antivirus software evaluation criteria. This study also examines how dynamic and manual analysis methods may affect detection rates, varying depending on file operational conditions.

The current research is geared towards the development of a malware detection system that is more accurate. Additionally, the capability to identify zero-day vulnerabilities, which are the product of customised tools and encompass traditional security mechanisms, is being examined.

## 3. Methodology

The efficacy of conventional signature-based techniques is constrained when identifying novel and sophisticated forms of malware. The continual development of malware, coupled with its growing complexity, leads to changes in its behaviour, creating challenges for its identification through traditional means. The CNN-LSTM algorithm presents a highly adaptable and resilient method for conducting malware analysis in the given context. Using the LSTM component in this algorithm is highly advantageous for capturing extended dependencies in sequential data, a critical factor in comprehending malware behaviour. LSTM, or Long Short-Term Memory, is a computational model that effectively preserves and retrieves important data for a prolonged duration using memory cells and gates. This characteristic renders LSTM highly capable of comprehending the complex patterns demonstrated by malicious software. The algorithm's capacity to understand temporal relationships enables it to evaluate the changing patterns of malware and deliver precise forecasts. On the other hand, the convolutional neural network (CNN) part of the system is very good at pulling out useful information from raw input. Within the realm of behavioural malware analysis, the raw data frequently consists of many types of information, such as system events, network traffic, and API calls [17], that are directly associated with the malware's behaviour. The CNN module utilises filters and convolutions on the input data, allowing it to identify spatial patterns and extract significant characteristics relevant to malware detection. The

ability to remove features boosts the algorithm's effectiveness in distinguishing behavioural patterns that differentiate malware from standard software.

One significant benefit of utilising the CNN-LSTM algorithm is its efficacy in identifying zero-day malware [18]. Zero-day malware refers to harmful software that exploits vulnerabilities that are either unidentified or do not have available patches. Conventional approaches that rely on established patterns or signatures face challenges in detecting zero-day malware, as they are insufficient for addressing these novel and previously unseen threats. On the other hand, the CNN-LSTM algorithm can thoroughly examine the behaviour of malware and identify anomalies or patterns that differ from expected norms. A behavioural approach enhances the algorithm's ability to detect previously unknown malware strains accurately. Moreover, the algorithm's capacity to evaluate temporal correlations makes it well-suited for identifying polymorphic and metamorphic malware. Polymorphic malware, as described in the literature [19], can modify its structure or visual characteristics while maintaining its destructive behaviour. This characteristic presents a significant obstacle for static analysis methods. Metamorphic malware extends this concept by actively altering its code to elude detection. The CNN-LSTM algorithm can identify enduring patterns across multiple varieties of malware, even when there are changes in static properties, by considering the behavioural aspects of the infection. The capabilities above allow the algorithm to identify polymorphic and metamorphic malware efficiently.

An additional noteworthy advantage of the CNN-LSTM algorithm is its proficiency in handling large quantities of data. The need for scalable analysis approaches becomes more evident as the number of malware samples rapidly increases and their behaviour becomes more sophisticated. The approach effectively integrates the parallel processing capabilities of convolutional neural networks (CNNs) with the memory-efficient characteristics of long-short-term memory (LSTM) models. The integration of these components allows the algorithm to handle datasets of considerable magnitude effectively. The significance of scalability is particularly apparent in cases involving real-time or near-real-time analysis, where the capacity to make timely decisions is crucial in preventing the spread of malware and minimising possible damage.

### *3.1. Hybrid Analysis using API calls*

During the investigation, Windows API calls were utilised within the executable to analyse the actions and characteristics of malicious software. The utilisation of Windows API calls is essential in dynamic malware research, as it offers valuable insights into the behaviour and objectives of malicious software[20]. Analysts and security researchers utilise API calls to enhance their comprehension of malware operations, detect their existence, and formulate efficient strategies for mitigating their impact. Examining the API calls performed by malicious software during its functioning enables evaluating its activities in a live context, providing crucial observations into its functionalities, such as the unauthorised extraction of data, the elevation of privileges, and the infiltration of computer systems. The main benefit of utilising API calls for dynamic malware analysis resides in their ability to comprehensively understand a malware's behaviour without requiring access to the underlying source code. It provides analysts with valuable information regarding the actions of both familiar and unfamiliar malware specimens, allowing them to detect any indications of unauthorised access. In addition, the analysis of API calls (as discussed in [21]) enables the categorization and classification of malicious software, hence enhancing the development of more efficient systems for detecting and preventing such threats.

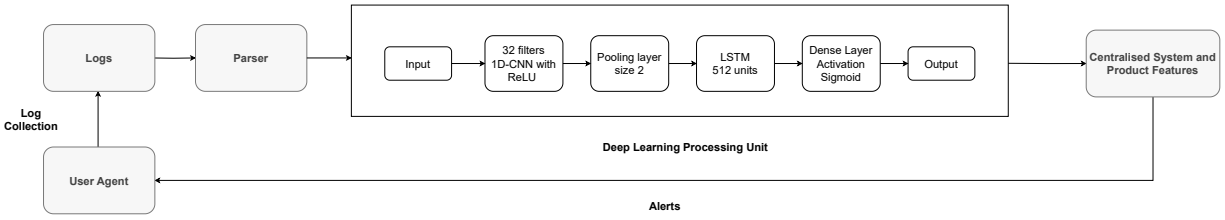


Fig. 1. Proposed Architecture diagram

The above diagram presented in Fig.1 describes the complete flow of the proposed system, where the deep learning part comprises the CNN-LSTM hybrid detection. The API logs used in this study were obtained via the Rohitab API monitor[22]. These logs were then translated into XML files retrieved from the API name tags and saved in a CSV

file. The provided CSV file has been meticulously annotated with the call sequence and functions as a fundamental reference point for further analytical procedures. To optimise the analysis procedure, the CSV file is associated with integer values derived from a thorough list of 308 distinct Windows API calls. The mapping process in this context refers to creating a structured representation of the API calls. The outcome of this mapping is a CSV file that contains the sequences that have been mapped. This CSV file serves as a input for deep learning models used in the research. Using deep learning in this situation is mostly about accurately predicting the next set of API calls[23]. The successful accomplishment of this objective enables the deep learning model to categorise the analysed programme as either malicious or genuine. When deciding on the most suitable machine learning model for this particular task, it is crucial to consider the potential obstacles and drawbacks, as these decisions substantially influence the model's effectiveness and accuracy. Therefore, careful consideration is required while selecting a machine-learning model for this categorisation assignment.

The combination of deep learning and API call analysis presents many advantages in the proactive detection of malware threats. Deep learning models can pick up complex relationships and patterns, which is impossible with traditional signature-based or heuristic detection methods. It demonstrates exceptional proficiency in differentiating between valid and malicious sequences of API calls, hence improving the accuracy of malware detection. Moreover, deep learning models exhibit the capacity to generalise from acquired patterns, therefore facilitating the detection of previously unknown security vulnerabilities and adapting to the swiftly changing landscape of malicious software. These models also streamline the process of extracting features from sequences of API calls, hence automating the discovery of key elements of harmful behaviour. By not requiring analysts to do time-consuming feature engineering, this method makes it easier to adapt to new malware techniques and is more effective at doing so. The caliber and variety of the training data significantly impact the effectiveness of deep learning in predicting malware from API requests. Ensuring the currency of a dataset that accurately reflects prevailing malware trends and evasion strategies is of utmost importance.

Overfitting is a problem that needs to be thought about in this framework. It happens when the model becomes too specific to the training data, which means it does not work well with new examples. Overfitting may occur when a model demonstrates excessive complexity or when the training dataset is insufficient in size or quality. To address the potential overfitting issue, it is advisable to employ techniques such as cross-validation and regularisation. Cross-validation is a technique that entails dividing the dataset into multiple subsets. The model is then trained on some subsets, while its performance is evaluated on the remaining subsets. This process allows for assessing the model's capacity to generalise to new data. Regularisation techniques are employed to apply restrictions on the parameters of a model to mitigate the risk of excessive complexity. Evaluating several machine learning models is essential when determining the most appropriate model for a given task. Models such as long short-term memory (LSTM) networks, gated recurrent units (GRUs), and recurrent neural networks (RNNs) have demonstrated potential for effectively addressing sequence prediction tasks. These models are highly suitable for analysing API call sequences and identifying trends indicative of malicious software behaviour. It is essential to use a diverse and representative dataset during the training phase. This process will make the model more valuable in real life and lessen the problem of overfitting. The dataset should include malware and benign software to ensure the model accurately differentiates between the two groups. In addition, optimising the model's performance necessitates the adjustment of hyperparameters, including the learning rate, batch size, and network design. The standard procedure encompasses using methodologies such as grid search and Bayesian optimisation to ascertain the most effective amalgamation of hyperparameters.

### *3.2. Dataset*

The dataset employed in this study was obtained from two main sources: "MALWARE ANALYSIS DATASETS: API CALL SEQUENCES," authored by Angelo Oliveira [24], and "MalbehavD-V1: A Dataset of API Calls Extracted from Malware and Benign Executable Files in Windows" [25]. The initial dataset is a deep learning-based malware detection and classification research component. It comprises 1,079 benign API calls and 42,797 sequences of malware API calls. An API call sequence is generated by extracting the initial one hundred consecutive, non-repetitive API calls linked to the parent process from the 'calls' section of Cuckoo Sandbox reports. Using the dynamic malware analysis methodology, MalBehvaD-V1 is a new dynamic dataset of API call sequences extracted from benign and malicious executables (EXE files) on Windows. The Cuckoo container was utilised to operate an isolated environment in which each file was executed. Malware samples were obtained from VirusTotal, whereas examples of benign software were

obtained from the CNET website. With a mixture of 1285 benign and 1285 malicious files, the dataset comprises 2570 files.

The number of malware samples in the previous dataset was significantly higher than that of benign samples, with a ratio of 40 to 1. The composition of the first dataset raised concerns about potential biases that could affect classification outcomes during the training phase. The initial dataset, "MALWARE ANALYSIS DATASETS: API CALL SEQUENCES," presented inherent biases that necessitated careful consideration. And to tackle such biases, the API-call-sequence dataset was incorporated. In contrast, the latter dataset was carefully designed to emphasise API calls, which function as textual representations of interactions with the operating system. The API call strings were methodically converted into numbers to improve the analytical process. To do this, a meticulous compilation of 308 unique API calls was conducted, wherein each call was systematically assigned a matching index. A Python script was developed to automate mapping API calls to their corresponding integers. Angelo Oliveira provided a prepared list, the basis for this script.

Following this, the two datasets were merged, resulting in a comprehensive dataset consisting of 2,500 instances of malware samples and 1,000 instances of benign samples. The deliberate method of harmonising datasets was implemented to address the bias concerns noted earlier and achieve a fairer representation of malware and benign executables. By utilising the information Angelo Oliveira provided, the research endeavour successfully curated a comprehensive compilation of API call sequences derived from a wide range of malware specimens. Incorporating a wide range of malware samples into the dataset significantly strengthens the analysis's strength and improves the results' applicability. In addition, adding executable files that are not harmful makes the dataset better by making it easier to compare malicious and benign software.

The dataset[1] assembly process was characterised by a rigorous methodology, with careful consideration given to every detail to maintain the integrity and quality of the dataset. Precautions were included to mitigate the occurrence of sample duplication and to facilitate a fair allocation of both malicious and non-malicious files. The rigorous approach employed in this methodology guarantees a reliable basis for the construction and evaluation of categorization models. Researchers and practitioners have the opportunity to utilise the comprehensive dataset provided to investigate a wide range of machine-learning techniques and algorithms to develop efficient models for the detection and classification of malware. The large number of malware samples and their clean counterparts in the dataset make it possible to get a full picture of how well the classification models work. In addition, the process of converting API call sequences into integers enhances the usefulness of the dataset, enabling the use of various statistical and mathematical methods. Using a numeric representation streamlines the feature extraction procedure and conforms to the prerequisites of algorithms that necessitate numerical input.

## 4. Experimental Setup

In order to replicate malicious actions, a custom script was created. The present script is designed to produce Sysmon logs, which are subsequently analysed using pre-established correlation criteria. Upon identification of a match, the logs are assigned a severity value determined by the criticality of the detected activity. In addition, all API calls made during this simulation are recorded and turned into integers, which means they can be used to train a CNN-LSTM model. The dataset consists of 175 columns, each representing a consecutive API request. Just like the correlation rules, these API calls are used to initiate alerts and determine risk scores according to their level of severity. The primary element of this system is a log parser analyzer implemented in Win32 API C++. It functions as a service. The above-mentioned programme is designed to analyse and interpret Sysmon logs, subsequently populating a database with the extracted information. In order to conduct a more comprehensive analysis of the data, a Python script is used to execute queries on the database and subsequently export the obtained findings to a CSV file.

The Rohitab API Monitor tool is used to capture API calls, extract requests, and process them using a Python script for XML parsing. The API calls are assigned integer values within a list of 308 distinct calls, which are then used as input for a CNN-LSTM model. The model uses a combination of convolutional neural networks and long short-term

[1] https://github.com/zarganaut/API-call-sequences

memory networks to analyse API call sequences, categorise activities, and generate a score. This provides valuable insights for making informed decisions and implementing appropriate actions.

The log parser analyzer, implemented using the Win32 API in the C++ programming language, functions as a fundamental module responsible for parsing Sysmon logs and facilitating the effective storage of data in a database. Correlation rules serve to augment the detection capabilities of a system through the activation of alerts and the assignment of severity scores to recognised threats. API monitoring is a process that gathers and records all pertinent API calls for subsequent analysis. The provided data is subsequently processed and converted into a format suitable for the CNN-LSTM model. By training the model using this dataset, it enables the machine to acquire knowledge of patterns and then make precise predictions about the characteristics and intensity of the identified activity.

The system's output consists of a categorization and a corresponding score, which may be used to make educated judgements regarding response actions. The implementation of suitable strategies can be undertaken to reduce risks and safeguard the system from potential harm, contingent upon the level of severity posed by the danger. In order to facilitate the efficient training and evaluation of the model, it was imperative to utilise a system equipped with graphics capabilities. The machine exhibited impressive technical specifications, comprising the Nvidia RTX 3050 graphics card, an AMD Ryzen 5 processor, 16 GB of RAM, and a 500 GB SSD. The inclusion of this robust configuration offered improved visual processing and computational capacities, hence yielding advantageous outcomes for the model. The utilisation of the CUDA framework was implemented to enhance the efficiency and performance of the training process. The CUDA platform, developed by Nvidia, is an application programming interface (API) and parallel computing framework that aims to leverage the computational capabilities of Nvidia GPUs for a wide range of general-purpose computing tasks, such as constructing machine learning models. The model took around 20 minutes to complete processing and generate the results.

The system utilises a CNN-LSTM model, which is characterised as follows:

1. *Data Partitioning* : The dataset is partitioned into separate training and testing subsets, utilising the train-test-split tool provided by the scikit-learn library. The data provided as input is commonly referred to as used data, and the variable "malware" is used to indicate the target variable associated with it. A test size of 0.25 is designated, signifying that a quarter of the data is reserved for testing purposes. The dataset undergoes a shuffling process, and a specific random seed value of 42 is assigned in order to ensure the reproducibility of the results. The resulting divisions are allocated to X-train, X-test, Y-train, and Y-test.
2. *Model Architecture*: The model is developed via the Keras Sequential API and consists of the subsequent layers:

   *Embedding Layer* : The embedding layer is responsible for transforming the input into dense word vectors, which have a dimensionality of 8. The input dimension refers to the total count of distinct API requests, which is 308 in this specific case. On the other hand, the input length is determined by the sequence length of the X-train data.

   *Batch Normalisation*: The Batch Normalisation Layer is responsible for normalising the activations of the preceding layer in order to provide stable training. Batch normalisation is an advantageous technique in CNN-LSTM architectures owing to its capacity to accelerate convergence, stabilise training dynamics, and mitigate internal covariate shifts. The layer normalises layer inputs in a CNN-LSTM model that combines convolutional and recurrent layers. This process ensures that the inputs remain in a consistent distribution during training, thereby mitigating problems such as gradients that evaporate or explode. This facilitates improved generalisation by serving as a form of regularisation in addition to accelerating training. The layer is a highly effective instrument for enhancing the consistency and resilience of the training procedure in intricate architectures such as CNN-LSTM. Consequently, it substantially contributes to the overall performance and robustness of the model.

   *Convolutional Layer*: The Conv1D layer is configured with 32 filters, each having a kernel size of 9 and utilising the 'relu' activation function. The padding parameter is assigned the value 'same' in order to ensure that the length of the output remains unchanged.

   *MaxPooling* : The MaxPool1D layer performs max pooling operations with a pool size of 2, resulting in a reduction in spatial dimensions.

   *LSTM* : The LSTM layer is comprised of 512 LSTM units and incorporates a dropout rate of 0.2. The value assigned to the return sequences parameter is false, which signifies that the LSTM layer will only return the final output.

*Dense Layer* : The dense layer is a type of fully connected layer that consists of a single unit and utilises the sigmoid activation function. It is commonly employed in binary classification tasks.

3. *Model Compilation* : The model is created utilising the Adam optimizer, with a learning rate of 0.0001. During the training process, the loss function is specified as 'binarycrossentropy', and the accuracy metric is calculated.
4. Model Training: The model undergoes training by utilising the fit approach, which involves the utilisation of the training data (X-train and Y-train). A validation split of 0.2 is utilised, whereby 20% of the training data is allocated for the purpose of validation. The training process spans 150 epochs, with a batch size of of 512. The training progress and performance metrics are stored in the history variable.

## 5. Results and Analysis

A Python script was used to retrieve data from the database that stores logs generated by the log-parser-analysis program. The script functioned based on pre-established correlation rules, generating alerts upon detecting correlations, indicating the presence of possible harmful behaviour. In addition, the logs were utilised in order to generate a CSV file that is appropriate for training machine learning models that are specifically designed for predicting malware. The deep learning CNN-LSTM model successfully categorised API calls exported from benign applications as benign and identified malicious API calls as malicious. When subjected to a randomly chosen dataset consisting of both malicious and benign samples, the model demonstrated a noteworthy validation accuracy of 96%. The aforementioned high level of accuracy serves to highlight the efficacy of the CNN-LSTM model in properly distinguishing between benign and malicious API calls.

The API requests, which were originally formatted in XML, were then converted into a CSV file, where the data is now represented as numbers. In order to achieve this objective, the API calls were systematically organised and categorised using a comprehensive list consisting of 308 distinct API call names. The CSV file that mapped these API calls was used as input for both the training and testing of the CNN-LSTM model. The establishment of correlation rules led to the activation of warnings whenever particular sequences of API calls were detected. The utilisation of this mapping procedure allowed the model to operate using numerical representations of the API calls, hence facilitating the processing and analysis processes.

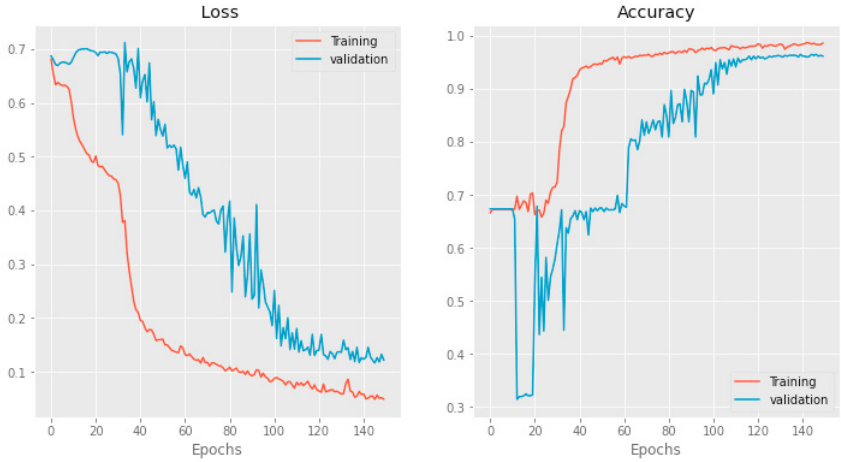


Fig. 2. Accuracy and loss

The diagram presented in Fig.2 illustrates the relationship between the number of epochs and the corresponding metrics of loss and accuracy throughout the training phase of the model. The data clearly demonstrates that with an increase in the number of epochs, a discernible trend emerges: the loss exhibits a decrease, while the accuracy demonstrates an increase. This graph indicates that the model demonstrated improved performance as time progressed. The decrease in loss suggests that the model has effectively minimised prediction mistakes, while the increase in

accuracy demonstrates its enhanced ability to generate precise classifications. These observations are consistent with the expected behaviour of a proficiently trained model. The graph provides visual evidence of the model's progress in learning and validates the effectiveness of the selected training methodology.

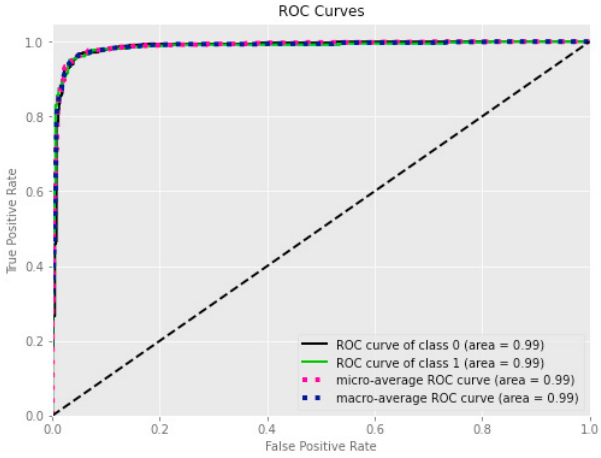


Fig. 3. ROC curve

The Receiver Operating Characteristic (ROC) curve utilises a two-dimensional visualisation, with the x-axis representing the false positive rate (FPR) and the y-axis representing the true positive rate (TPR). An optimal curve would closely follow the upper-left region of the graph, demonstrating a high true positive rate (TPR) and a low false positive rate (FPR), which is suggestive of a classification model that performs well. The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is a widely employed statistic for evaluating the overall performance of a model. A receiver operating characteristic (ROC) value of 1 indicates an ideal classifier, whereas a value of 0.5 implies a classifier that exhibits performance equivalent to random chance (Fig. 3). The closeness of the AUC-ROC value to 1 indicates the model's capacity to discriminate between the two distinct groups accurately.

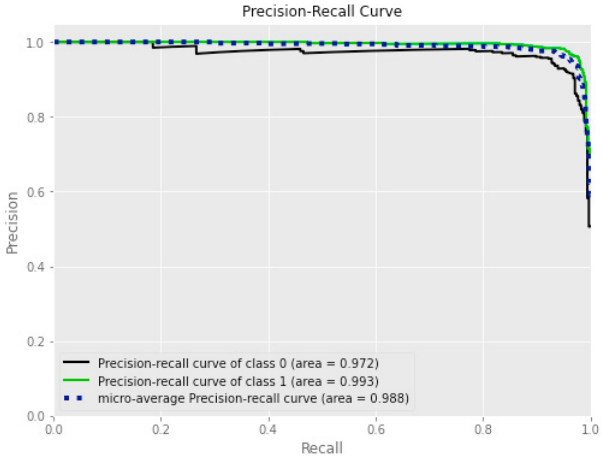


Fig. 4. Precision Recall Curve

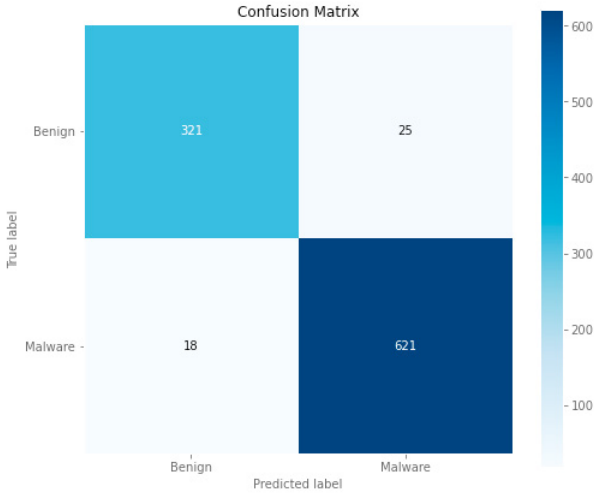


Fig. 5. Confusion Matrix

A model that exhibits high precision aims to reduce the occurrence of false positive predictions, whereas a model with high recall strives to capture the bulk of positive examples. The optimal situation entails a curve that closely adheres to the upper-right corner of the graph, indicating the simultaneous presence of elevated precision and recall. Nevertheless, it is common to encounter a compromise between precision and recall, as improvements in one metric might potentially lead to negative consequences for the other. The Precision-Recall curve, akin to the ROC curve, is assessed by means of the Area Under the Curve (AUC-PR). A higher area under the precision-recall curve (AUC-PR) signifies the better performance of the model, where a perfect classifier achieves an AUC-PR value of 1, as observed in this particular scenario (Fig. 4). The confusion matrix (Fig. 5) shows that the True positive and True negatives are highest indicating the classification is correct and can also classify new data correctly. The accuracy metrics were as follows

CNN_LSTM model classification report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.93 | 0.94 | 346 |
| 1 | 0.96 | 0.97 | 0.97 | 639 |
| accuracy | | | 0.96 | 985 |
| macro avg | 0.95 | 0.95 | 0.95 | 985 |
| weighted avg | 0.96 | 0.96 | 0.96 | 985 |

Fig. 6. Classification report

## 6. Conclusion

In conclusion, the ever-evolving strategies of malware authors pose persistent challenges for conventional detection methods, necessitating a shift towards behavioural analysis and advanced deep learning models. While signature-based approaches have been effective against known threats, their limitations become evident when faced with novel and sophisticated malware variants. Discrepancies among antivirus providers further underscore the need for more reliable and adaptive cybersecurity solutions. This research contributes by presenting a comprehensive system for detecting and analysing malicious activity. The CNN-LSTM model's exceptional performance in classifying API calls strengthens the system's capabilities. Training on diverse datasets equips the model to accurately categorise new API calls as benign or malicious, providing valuable insights for threat detection and response in a dynamic threat landscape.

However, it's important to acknowledge limitations and consider future directions. Limitations include the need for substantial computational resources, which may pose challenges for smaller organisations. Future work could focus on optimising model efficiency and exploring real-time threat detection capabilities. Additionally, further research into countering adversarial attacks on deep learning models is essential in enhancing the system's robustness. In this ever-changing cybersecurity landscape, embracing behavioural analysis and advanced deep learning models exemplifies the commitment to staying resilient against evolving threats. By addressing limitations and pursuing future research avenues, the enhancements to the cybersecurity defences can be continued and adapted to emerging challenges effectively.

## References

[1] J. Singh, J. Singh, A survey on machine learning-based malware detection in executable files, Journal of Systems Architecture (2021). URL https://www.sciencedirect.com/science/article/pii/S1383762120301442

[2] A. Sidhardhan, S. Keerthana, J. M. Kannimoola, Weaponizing real-world applications as c2 (command and control), in: 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), IEEE, 2023, pp. 458–463.

[3] R. Sihwail, K. Omar, K. A. Zainol Ariffin, A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis 8 (2018) 1662. doi:10.18517/ijaseit.8.4-2.6827.

[4] A. Aslan, R. Samet, A comprehensive review on malware detection approaches, IEEE Access 8 (2020) 6249–6271. doi:10.1109/ACCESS.2019.2963724.

[5] A. V, V. P, V. G. Menon, A. K. E R, A. Shilesh, A. Viswam, A. Shafiq, Malware detection using dynamic analysis, in: 2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS), 2023.

[6] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, S. Venkatraman, Robust intelligent malware detection using deep learning, IEEE Access 7 (2019) 46717–46738. doi:10.1109/ACCESS.2019.2906934.

[7] J. Zhang, Deepmal: A cnn-lstm model for malware detection based on dynamic semantic behaviours, in: 2020 International Conference on Computer Information and Big Data Applications (CIBDA), 2020, pp. 313–316. doi:10.1109/CIBDA50819.2020.00077.

[8] H. Daku, P. Zavarsky, Y. Malik, Behavioral-based classification and identification of ransomware variants using machine learning, in: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018, pp. 1560–1564. doi:10.1109/TrustCom/BigDataSE.2018.00224.

[9] X. Yuan, Phd forum: Deep learning-based real-time malware detection with multi-stage analysis, in: 2017 IEEE International Conference on Smart Computing (SMARTCOMP), 2017, pp. 1–2. doi:10.1109/SMARTCOMP.2017.7946997.

[10] Q. Chen, R. A. Bridges, Automated behavioral analysis of malware: A case study of wannacry ransomware, in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017.

[11] M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song, C. Park, Flow-based malware detection using convolutional neural network, in: 2018 International Conference on Information Networking (ICOIN), 2018.

[12] A. O. Prokofiev, Y. S. Smirnova, V. A. Surov, A method to detect internet of things botnets, in: 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018.

[13] A. Dhammi, M. Singh, Behavior analysis of malware using machine learning, in: 2015 Eighth International Conference on Contemporary Computing (IC3), 2015.

[14] M. Al-Janabi, A. M. Altamimi, A comparative analysis of machine learning techniques for classification and detection of malware, in: 2020 21st International Arab Conference on Information Technology (ACIT), 2020.

[15] P. V. Dinh, N. Shone, P. H. Dung, Q. Shi, N. V. Hung, T. Nguyen Ngoc, Behaviour-aware malware classification: Dynamic feature selection, in: 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 2019.

[16] A. Zarghoon, I. Awan, J. P. Disso, R. Dennis, Evaluation of av systems against modern malware, in: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), 2017.

[17] H. Zhang, X. Yun, X. Deng, X. Zhong, Malware classification based on gaf visualization of dynamic api call sequences, in: 2022 IEEE 22nd International Conference on Communication Technology (ICCT), 2022, pp. 1873–1877. doi:10.1109/ICCT56141.2022.10073202.

[18] H. Al-Rushdan, M. Shurman, S. H. Alnabelsi, Q. Althebyan, Zero-day attack detection and prevention in software-defined networks, in: 2019 International Arab Conference on Information Technology (ACIT), 2019, pp. 278–282. doi:10.1109/ACIT47987.2019.8991124.

[19] I. You, K. Yim, Malware obfuscation techniques: A brief survey, in: 2010 International Conference on Broadband, Wireless Computing, Communication and Applications, 2010, pp. 297–300. doi:10.1109/BWCCA.2010.85.

[20] W. R. Aditya, Girinoto, R. B. Hadiprakoso, A. Waluyo, Deep learning for malware classification platform using windows api call sequence, in: 2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS, 2021, pp. 25–29. doi:10.1109/ICIMCIS53775.2021.9699248.

[21] O. Hachinyan, Detection of malicious software on based on multiple equations of api-calls sequences, in: 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017, pp. 415–418. doi:10.1109/EIConRus.2017.7910580.

[22] M. F. Marhusin, H. Larkin, C. Lokan, D. Cornforth, An evaluation of api calls hooking performance, in: 2008 International Conference on Computational Intelligence and Security, Vol. 1, 2008, pp. 315–319. doi:10.1109/CIS.2008.199.

[23] S. Alqurashi, O. Batarfi, A comparison between api call sequences and opcode sequences as reflectors of malware behavior, in: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), 2017, pp. 105–110. doi:10.23919/ICITST.2017.8356357.

[24] [link].
URL https://ieee-dataport.org/open-access/malware-api-call-dataset

[25] Mpasco, Mpasco/malbehavd-v1: Public datasets of malware and benign executable files (windows exe files). the dataset can be used by cybersecurity researchers focusing on the area of malware detection. it is suitable for training and testing both machine learning and deep learning algorithms.
URL https://github.com/mpasco/MalbehavD-V1