Journal of Information Systems Engineering and Management
2025, 10(21s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Cutting-Edge Deep Learning Framework for Streamlined Malware Detection in Cybersecurity Utilizing Long Short-Term Memory (LSTM)

[1.] Mrs. A. Anuradha, [2.] Dr. Arun Singh Chouhan, [3.] Dr. S Srinivas Rao

[1]*Research Scholar, Department of CSE, School of Engineering, Malla Reddy University, Hyderabad, India. Email-ID: 2232CS020001@mallareddyuniversity.ac.in*

[2.] *Associate Professor, Department of CSE, School of Engineering, Malla Reddy University, Hyderabad, India.*

Email-ID: arunsingh@mallareddyuniversity.ac.in

[3.] *Principal and Professor, Department of ECE, School of Electronics, Malla Reddy College of Engineering and Technology, Hyderabad,India. Email-ID: ssrao.atri@gmail.com*

ARTICLE INFO

Received: 20 Dec 2024

Revised: 02 Feb 2025

Accepted: 18 Feb 2025

ABSTRACT

Malware is a malicious piece of code that has been causing security issues to information systems and networks across the globe. The entire cyberspace is suffering security issues due to malware. The traditional approaches based on heuristics could provide specific mechanisms to protect information systems from malware. However, the ever-increasing malware patterns made it very difficult to detect new malware with the help of heuristics-based methods. The emergence of Artificial Intelligence (AI) has paved the way for learning-based approaches that could provide better performance in malware detection. However, Enhancements are required to the current deep learning techniques utilized in malware detection in addition to updating trading data used for the supervised learning process. Our proposal in this study was a deep learning framework for efficient malware identification in cybersecurity with the use of Long Short Term Memory (LSTM). Our approach, which we called Learning-Based Malware Detection (LBMD), which exploits the efficient dataset created as part of the framework and uses the LSTM model appropriately to detect malware efficiently. Our empirical study with the dataset created reveals that the proposed algorithm outperforms many existing malware detection models like KNN, XGBC, and baseline CNN with the highest accuracy of 98.79%.

Keywords: Cyber Security, Malware Detection, Deep Learning, Artificial Intelligence, Long Short Term Memory.

INTRODUCTION

Software designed specifically to harm networks and information systems is known as malware causing severe security issues. Any software is considered malware if it has a secret agenda that is against the interests of the application user. The targets of malware include personal computers, smartphones, application servers, and other electronic devices. The development of malware has been increasing with diversified architectures and approaches, causing alarming security situations as it actually is [1]. Together with the proliferation of smartphone app development platforms, mall wave developers have also targeted mobile applications. Android and iOS platforms have found issues with malware in their mobile applications used by smartphone users. Several frameworks are used to detect malware in Android platforms [2].

Many contributions are found in the literature to detect malware automatically and on different platforms. The hybrid CNN-LSTM model attains high accuracy in threat detection across various datasets [6]. AOAEL-CDC is the approach used with ensemble learning. Future priorities will include standardized protocols and adaptive security [7]. The complexity of malware is evolving, challenging conventional detection techniques. AAMD-OELAC uses DL and ensemble learning to identify Android malware better than other approaches [10]. Increased cyber-attacks necessitate more sophisticated security measures. Mining Cyber Threat Intelligence (CTI) yields insightful information for pre-emptive security [15]. From the literature, it was observed that DL models are widely used for malware detection research. However, there is a need to improve performance in leveraging dataset creation, preprocessing, and supervised learning mechanisms. The following are the things we contributed to this paper.

1. We put forth a deep learning architecture utilizing Long Short Term Memory (LSTM) for effective virus detection in cybersecurity.

2. To effectively identify malware, we devised a method called Learning-Based Malware Detection (LBMD), which makes use of the effective dataset built within the framework and suitably applies the LSTM model.

3. using the dataset created and a prototype application, our empirical study reveals that the proposed algorithm outperforms many existing malware detection models.

This is how the rest of the paper is organized. The literature is reviewed in Section 2 and includes many research on malware detection. Section 3 provides the background data needed to comprehend the suggested technique. The suggested deep learning framework for effective malware detection is presented in Section 4. The findings of our empirical investigation using the Windows API calls dataset are shown in Section 5. Our research is concluded in Section 6 along with recommendations for more investigation.

2. RELATED WORK

Numerous deep learning-based techniques are now in use for malware identification. Karat et al. [1] maintained robust cybersecurity; dynamic detection using CNN-LSTM is becoming more critical due to the increasing strategies used by malware developers to circumvent static analysis. Smmarwar et al. [2] increased the proliferation of smart gadgets and internet services. Critical components for detection include DL models, ML, and anti-malware software. Bensaoud et al. [3] explored DL techniques, such as text/image classification and countering adversarial assaults, which are necessary due to the complexity of malware detection. Galli et al. [4] investigated AI-based solutions prompted by malware threats. XAI techniques improve deep learning model comprehension and assessment. Devi et al. [5] integrated into intelligent cities for efficiency. However, there are cybersecurity threats. By utilizing DL methods and optimization algorithms, the FOADL-EMAR approach improves the identification of harmful behavior.

Nazir et al. [6] expanded so fast that strong cybersecurity is required. The hybrid CNN-LSTM model attains high accuracy in threat detection across various datasets. Allafi et al. [7] prevented illegal access and network security breaches—the accuracy of the AOAEL-CDC approach with ensemble learning. Future priorities will include standardized protocols and adaptive security. Mbunge et al. [8] increased cyberattacks result from rising smartphone usage. Hence, Android apps must use efficient malware detection techniques like deep learning models. Although crucial, Apruzzese et al. [9] state that machine learning is only partially employed in cybersecurity. The advantages, difficulties, and uses of machine learning are summarized in this article. Alamro et al. [10] lived in virtual worlds due to technological improvements. The complexity of malware is evolving, challenging conventional detection techniques. AAMD-OELAC uses DL and ensemble learning to identify Android malware better than other approaches.

Ahmed et al. [11] rapid evolution of malware motivates machine learning research in network security. High-accuracy malware categorization strongly suits deep learning, mainly transfer learning on CNN. Keshk et al. [12] suggested an XAI architecture that achieves excellent accuracy and interpretability for IoT intrusion detection using LSTM. Lakshmi and Karthika [13] provided security dangers in addition to benefits. An innovative DCCNN-SMO model effectively identifies malware and piracy. Santhadevi and Janet [14] increased IoT devices present security vulnerabilities that cybercriminals prey on. Edge Service uses deep learning to improve malware detection. Sun et al. [15] increased cyber-attacks necessitate more sophisticated security measures. Mining Cyber Threat Intelligence (CTI) yields insightful information for pre-emptive security.

Macas et al. [16] presented viable answers to cybersecurity problems in the face of growing data quantities and changing threats. Chaganti et al. [17] increased the number of IoT devices vulnerable to virus breaches because of default settings. High accuracy is achieved with a Bi-GRU-CNN DL model. Muthanna et al. [18], though profitable, there are security dangers associated with IoT. Outperforming benchmarks, a cuLSTMGRU-based SDN framework detects threats with colossal accuracy. Pawlicki et al. [19] discussed the difficulties in putting Artificial Neural Networks (ANNs) into practice in cybersecurity, particularly for IDS. Solutions for dataset balance, adversarial attack management, and hyperparameter tweaking are covered. It also looks at moral and societal problems. Capuano et al. [20] challenged AI's ubiquitous yet opaque use in daily life. Although it presents security problems, explainable AI (XAI) gives insights. The role approaches and difficulties of XAI in cybersecurity are investigated.

Kimmell et al. [21] found cloud services to be more and more essential, but there are security risks, particularly from malware. LSTMs and BIDIs are examples of recurrent neural networks (RNNs) that exhibit above 99% efficacy in malware detection. Performance is unaffected by input order, but training time is—plans for the future call for more extensive testing. Haq et al. [22] have become increasingly critical for organizations, but there are security risks, particularly from malware. Training time is impacted by input sequence, but performance is not. Larger-scale tests are planned for the future. Jahromi et al. [23] needed to detect malware because threats are evolving quickly. Real-time detection is enhanced, particularly in critical systems, by a pre-trained stacked LSTM because of its increased accuracy and convergence. Bibi et al. [25] widespread use of Android draws sophisticated malware. A malware detection system based on GRU strengthens defenses against Android threats by achieving high accuracy.

Qureshi et al. [26] needed strong security measures since cyber threats worsen. An DL-based malware detection system achieves incredible accuracy. Kumar et al. [27] presented a severe risk in the digital sphere, necessitating the

development of efficient detection techniques. Deep learning performs better than traditional algorithms in malware detection, highlighting scalable frameworks for real-time deployment. Rodriguez et al. [28] covered detection techniques and performance assessment for contemporary deep learning applications in mobile network security. Deep Learning (DL) models are becoming more popular as traditional cybersecurity solutions fail to keep up with the growing complexity of threats. Qiu et al. [29] issued like data quality and architectural selection, DL transforms cybersecurity research, particularly in the area of Android trojans detection. Research investigates DL architectures such as FCN, CNN, RNN, DBN, AE, and hybrids to enhance the representation of code semantics. Shukla et al. [30] used many methods for malware detection, such as machine learning, deep learning, and vision. The two-pronged technique that has been proposed combines recurrent neural networks with micro architectural traces to identify classic and stealthy malware with excellent accuracy.

Fang et al. [31] complicated data patterns, using a BRNN-LSTM architecture improves cyber threat prediction accuracy beyond statistical methodologies. Amin et al. [32] increased malware that required sophisticated detection techniques. This study proposes a high-accuracy end-to-end deep learning system. Barut et al. [33] focused on analyzing accuracy and runtime performance while identifying encrypted malware using TLS through machine learning and deep learning methods. Li et al. [34] suggested using machine learning to find Domain Generation Algorithms (DGAs) essential for security. Mani et al. [35] suggested using Windows Performance Counters data to create a virus detection system based on deep learning solution for crypto mining called DeCrypto Pro.

Reference	**Approach**	**Technique**	**Dataset**	**Limitations**
[3]	DL	A fuzzy-import hashing technique	ImageNet, DAMD, and NSL-KDD	The misclassification issue is to be resolved.
[4]	Learning based approach	ML and DL techniques	Mal-API-2019, API Call Sequences, and Alibaba	XAI techniques and the robustness of adversarial architecture are yet to be explored.
[6]	ML and DL	CNN-LSTM hybrid	IoT23, N-BaIoT, and CICIDS2017	Adaptive learning approaches are to be integrated with their framework.
[7]	Machine Learning and FS approaches	AOAEL-CDC technique	UNSW-NB15	Context-aware and adaptive security approaches are to be explored.
[10]	AAMD-OELAC and the hunter-prey optimization (HPO) approaches	AAMD-OELAC technique	Andro-AutoPsy	The privacy-preserving technique has yet to be incorporated.
[18]	ML and DL	AI-based techniques	CICIDS2017	Hybrid DL models in IoT use cases are the possible future scope of the research.
[24]	ML	Machine learning techniques	Custom dataset	They intend to improve it to work with time series and text-based datasets.
[28]	DL	DBN technique	CICIDS2017, NSL-KDD and KDDCUP'99	Cost optimization and parallel processing are to be incorporated.
[34]	ML	DGA-based techniques	DGA datasets	Further investigation is desired on more optimizations with deep learning.

[37]	ML and DL	GRU, LSTM, CNN, CNN LSTM, and DNN.	KDDCUP99, UNSW-NB-2015, CICIDS-2017 and NSL-KDD	Parallel processing and unsupervised DL approaches are yet to be explored.

Table 1: Summary of findings in the literature

Mahdavifar and Ghorbani [36] examined deep learning (DL) use in cybersecurity, particularly emphasizing phishing/spam, malware, intrusion, and website defacement detection. Along with discussing model selection, future approaches, and adversarial assaults, it puts out a DL framework. Samy et al. [37] presented a DL-based IoT threat detection framework utilizing an LSTM model installed on fog nodes for edge-layer analysis. Compared to others, it does better DL models with a vast accuracy and high detection rate. With a refined BERT model, Yin et al. [38], the ExBERT framework predicts vulnerability exploitability with tremendous accuracy and high precision. Concept drift will be addressed, and multi-source aspects will be integrated into plans. Hasan et al. [39] proposed as part of an SDN-enabled control plane that can identify and neutralize rogue IoT nodes with colossal accuracy. Akarsh et al. [40] presented a deep learning approach that achieves immense accuracy in malware classification when applied to the Malimg dataset. Table 1 shows a summary of the findings of the literature. From the literature, it was observed that DL models are widely used for malware detection research. However, there is a need to improve performance in leveraging dataset creation, preprocessing, and supervised learning mechanisms.

3. PRELIMINARIES

This section presents the preliminaries required to understand the deep learning framework that has been suggested for effective malware detection.

3.1 Long Short Term Memory

Recurrent neural networks (RNNs) are the foundation of LSTM, a deep learning method [6]. The creation of LSTM occurred from RNN's inadequate efficacy for long-term learning. Random intervals can be used by the LSTM architecture to recall and learn any long-term reliance. When examining data or events that have a specific connection, especially chronology, it is seen to be a useful method, as discussed in [7]. T iteration has the following definition: if the time series data are, for example, $x = \{x_1,\cdots, x_T\}$, $h = \{h_1,\cdots, h_T\}$ is the vector sequence that is concealed and $y = \{y_1,\cdots, y_T\}$ displays an output vector sequence.

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = F(W_{hy}h_t + b_y) \quad (1)$$

The activation function is denoted by F, whereas the weight matrices for the connections throughout calculation time are shown by W_{xh}, W_{hh}, and W_{hy}. Where W_{xh}, W_{hh}, and W_{hy}Are weight matrices associated with the computation time relationship and FIs the activation function.

3.2 Virus Total

An internet tool called VirusTotal examines dubious files or URLs [4]. Various online browsers and antivirus program engines run dubious files or URLs to carry out hazardous activity. An extensive report including DNS resolutions, registry access, and other topics is provided by each antivirus program engine. and other details. The VirusTotal service offers an interface that is devoid of interpretation for analysis results obtained from antivirus software. Additionally, users can run fresh analyses and access other users' analysis results because this service has an extensive analysis archive. With the help of Virus Total's VirusTotalPublic API v2.0 interface, developers of applications may receive services without requiring a web browser and analyze files and URLs automatically [5]. Typically, the response's body is JSON, and the analysis findings from web browsers and antivirus program engines are separated. After analyzing the information we received from the API, we identified the malware families and provided labels for each piece of malware.

3.3 Cuckoo Sandbox

Compatible with several operating systems, the Cuckoo Sandbox program is a public and free application [2]. This tool may provide a complete analysis report on the files deemed suspicious [3] as part of malware studies. Malicious software may be compiled and executed in a setting akin to a workplace using the Cuckoo Sandbox program. Its functions include file analysis and gathering detailed analysis data on the structural characteristics and activity of malicious software, including memory dumps, network traffic, malware API calls, and more. A MongoDB database in JSON format contains the gathered data. There are two primary parts of Cuckoo Sandbox. The first part is the management computer, which launches the user-facing web service, begins the malware analysis, and stores the findings in the database. The analysis machine is the second part; it might be either real or virtual. This computer runs the malicious software and analyzes, simulating the malicious program's operating environment. We collect the Windows API call sequences that mimic malware activity using this software as part of our study.

3.4 Windows API Calls

Primarily intended for developer-OS interaction, Developing programs for the Windows operating system is possible with the help of the Windows API [1]. As a result, the OS provides a wide range of services, including API 2. To utilize a feature provided by the operating system, an application created for the Windows platform has to use the interfaces provided as APIs. Regardless of the operating system, an application must contact many APIs to finish an operation. For instance, Windows API 3's Create File function is triggered when an application requests to create a file. An application's entire behavior on the system may be seen in all of its API calls. As a result, the technique based on API calls is frequently used in dynamic malware research to illustrate the behavior of malware properly. We collect Windows API calls made to the operating system by malicious software and offer a feature set in this effort. Afterward, we train the classifier to identify malware using these properties.

4. MATERIALS AND METHODS

The two primary goals of this research are as follows: First, we generated a pertinent dataset; second, we used this dataset to conduct a comparison analysis employing different machine learning techniques to identify and categorize malware according to its categories automatically.

4.1 Problem Definition

Provided a malware sample, Creating an automated malware detection system using deep learning is a difficult task.

4.2 Dataset Creation

Data set creation is a critical process for leveraging the performance of wave learning models. This effort has produced several significant contributions, including malware analysis data and the latest sequencing dataset for Windows PE Malware API. In this collection, there are 7107 malware instances across several categories. As previously said, To determine the virus classifications, utilize the VirusTotal service and the Cuckoo Sandbox tool extracts harmful program call sequences for the Windows API. The system architecture utilized for data collection and LSTM algorithm classification is shown in Figure 1.

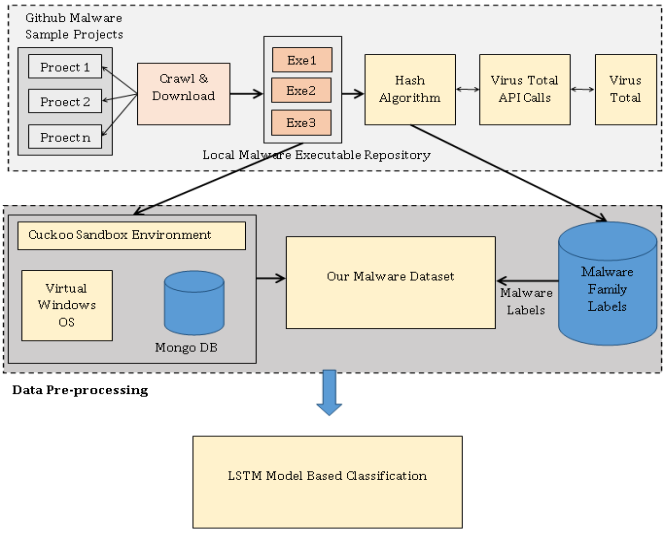

Figure 1: The proposed deep learning framework for efficient detection of malware

Categorization of data, pre-processing and analysis, and data collection are the three main parts of our system. To construct the dataset, follow the procedures given below. The Cuckoo Sandbox software is installed on a computer that is running Ubuntu Linux. Installing and examining malware was done by configuring the analysis system as a virtual server. Windows is the operating system that this server is set up with. It is not possible for malware to run since the firewall has been disabled without upgrading the operating system.VirtualBox 5.1.30, Windows 7 Analysis OS, Cuckoo2.0.4, Ubuntu 17.10, and Python 3.6.5 are the versions of the tools. After gathering the data, we use Cuckoo, a dynamic analysis engine, to analyze and pre-process the malware. More than twenty thousand malware instances have been run separately using the Cuckoo Sandbox tool. An entirely MongoDB database houses the results. We were able to get the malware's activity data on the analysis computer using this analysis information. All of the malware's Windows API call requests on the Windows 7 operating system correspond to this behavior. Among the pre-processing tasks for data are:

4.3 Methods

This subsection presents information about the proposal framework, which consists of preprocessing training and efficient malware detection.

4.3.1 Preprocessing

Upon closer inspection, We found that the Windows API calls in the dataset were unique. These API calls have an index between 0 and 341 in range. As such, each row in the collection represents the series of API calls needed to analyze malware. We changed the order in which the API calls were made and removed from the dataset any rows that did not contain ten or more different API calls as these malicious programs are specifically designed for a particular application. We ascertained the hash values of every malware sample we examined and consulted the Virus Total website for these numbers. The Virus Total service's analysis findings were saved in a database. Consequently, many antivirus engines reviewed each piece of malware, and the analysis's findings were noted. We have classified all of the malware based on the findings of every investigation that we were able to collect with the use of this service. In the process, we discovered that various antivirus programs return disparate findings for the same infection or that some malware may evade detection by antivirus software. Examining the hash value 06e76cf96c7c7a3a138324516af9fce8 on the Virus Total service, for example several antivirus programs identify this file as a worm, DrWeb as a trojan, and Babable as a clean file. As a result, we took the majority class in the analyses into account when identifying the malware classes. A sample is classified as positive if most engines agree it is harmful.

This study utilized The LSTM algorithm to construct the malware classification approach. Since it operates on sequential data, TF-IDF or any other vectorization model is not necessary for the LSTM approach to work. Therefore, it is essential to compare the classification performance of the suggested method with other traditional machine learning techniques, such support vector machines, decision trees, and k-nearest neighbor. Our dataset is suitable for these conventional classification techniques thanks to the TF-IDF model. We go over how to use the TF-IDF algorithm to convert the text in Section 3.2.

4.3.2 TF-IDF

A conventional numerical dataset generation The phrase "frequency-inverse document frequency-based text dataset vectorization" refers to this method. Based on the frequency with which a word appears in a document d, each term t is assigned a weight [8]. $tf_{(t,d)}$'s the process name for this one. The vector produced in this way might be considered a digital synopsis of a document. Assume that the raw frequency's f(t,d) represents t. The following is a list of term frequencies:

$$tf_{(t,d)} = f(t,d) \qquad (2)$$

On the other hand, every term in a dataset created solely based on phrase frequencies is given identical weight. The frequency at which a certain condition occurs in a collection is quantified by the inverse document frequency. The group's total number of documents, N, divided by the frequency of documents, or the logarithm of that division, $d\ f_t$ is the definition of the inverse document frequency (IDF).

$$id\ f_t = log\ log\ \frac{N}{d\ f_t} \qquad (3)$$

If the value of df_t is low in the documents in corpora, it leads to higher value of *idft*. Provided the high value of the frequency of dt_f, it leads to a low $id\ f_t$ value. After merging the term frequency, inverse document frequency (TF-IDF) matrix is formed. $tf_{t,d}$, with the opposite document frequency,$id\ f_{t,}$to determine the weights of the terms seen in each document.

$$tf - id\ f_t = tf_{t,d} \times id\ f_t \qquad (4)$$

The LSTM algorithm was used in this work to construct the malware classification approach. Since the LSTM method operates on sequential data, No vectorization model, like TF-IDF, is needed for it. The classification performance of the suggested method must be compared with that of other well-established machine learning techniques, including decision trees, support vector machines, and k-nearest neighbor. Since other classification algorithms can only handle numerical data, the TF-IDF model is the only one used. TF-IDF is not utilized in our model.

4.3.3 Classifier Learning

To construct classification models for eight different types of malware, steps 1 through 7 are carried out individually for each form of malware. Python programming and the machine learning tools Keras, Tensorflow, and Scikit-learn are used in the research. Using the Keras library, we created two-tier LSTM frameworks and built LSTM networks. We lay out the overall steps for our classifier creation step in Algorithm 1. Therefore, the time and space complexity of our method is O (n). For each unique label, a representative classifier is constructed using the main loop of the algorithm.

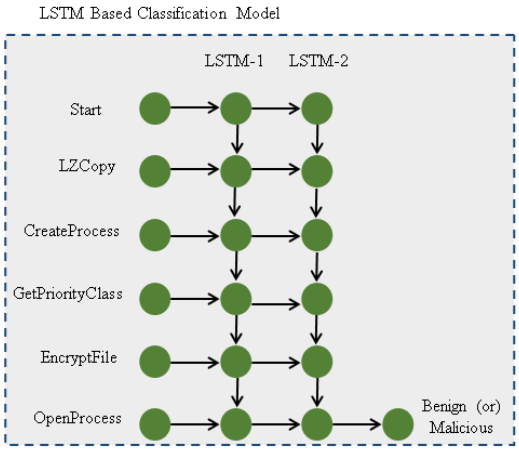

Figure 2: The LSTM-based classification model used in the proposed framework

Figure 2 shows the process of training. Each time a virus calls an API on Windows, the LSTM network model receives the call and gives y < the class label. We employ a binary classifier across all eight malware classes. The use of the log loss function as the loss function is seen in Equation 5.

$$l(y,\hat{y}) = \frac{1}{L}\sum_{l=1}^{l=|L|} -(y_l log(\hat{y}_l) + (1-y_l) log(1-\hat{y}_l)) \qquad (5)$$

The mythology for malware detection has certain important steps, as summarized here. We decide which virus class to categorize. For the selected malware kind, we prepare the dataset. Relevant malware-type information is labeled with a "1" by the model, whereas other categories are labeled "0." We define and design a two-tier LSTM-based classification model. 80 percent of the data are used to train the classifier during this phase. Throughout training, validation is conducted using twenty percent of the training data set. We test the trained classifier on a 20% sample of the dataset. This process displays the program's API calls to the classifier and applies a class label to newly created software instances based on the model's voting results. Documentation is done for training and test results. 80 percent of the data are used to train the classifier during this phase. Throughout training, validation is conducted using twenty percent of the training data set.

4.4 Dataset Details

The malware classifications and Windows API call sequences were matched by the labeled training dataset. Eight distinct malware classes are included in this dataset, which is accessible to the public on the GitHub website.

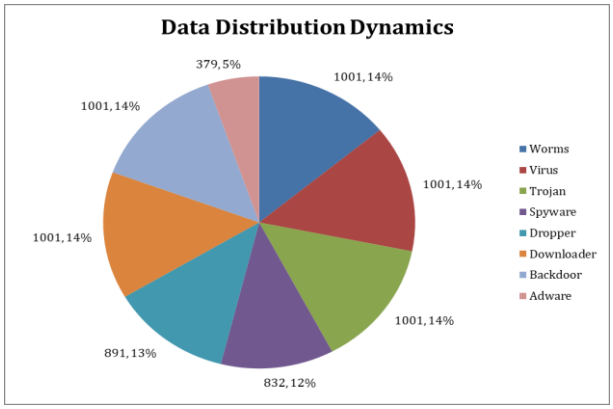

Figure 3: Data distribution dynamics in terms of class-wise number of instances and percentage of distribution

As presented in Figure 3, the dataset used in this research contains different classes of malware. Adware has 379 instances; backdoor has 1001 instances; downloader has 1001 instances; dropper has 892 instances; spyware has 832 instances; trojan has 1001 instances of virus, and Wong's has 1001 instances each.

4.5 Proposed Algorithm

An algorithm with a learning-based approach is proposed for efficient malware detection. The algorithm has proven effective in exploiting various aspects of malware-affected samples in supervised learning.

Algorithm: Learning-Based Malware Detection (LBMD)
Input: Malware dataset D, classes C, class labels L

Output: Malware detection results R, performance statistics P

1. Begin
2. D'☐Preprocess(D)
3. (T1, T2)☐SplitData(D')
4. For each c in C
5. $L_c \leftarrow \{1, \ if \ l = c \ 0, otherwise$ //label change
6. Build LSTM classification model m
7. Compile model m
8. m'☐TrainModel(T1, m)
9. R☐DetectMalware(T2, m')
10. P☐Evaluation(R, ground truth)
11. Print R for c
12. Print P for p
13. End For
14. End

Algorithm 1: Learning-Based Malware Detection (LBMD)

Algorithm 1, Learning-Based Malware Detection (LBMD)k, takes a malware dataset D as input and aims to classify malware into classes C using class labels L. Performance statistics P and malware detection findings R for each class constitute the algorithm's output. The malware dataset is first preprocessed, and the data is divided into training set T1 and test set T2. For each class c in C, the algorithm builds an LSTM classification model, compiles the model, trains it on the training set T1, detects malware in the test set T2, evaluates the detection results R is compared with the ground truth, and the performance statistics P and R for each class are printed. In a structured manner, the algorithm performs specific steps to achieve its objective of malware detection. It emphasizes the preprocessing of the dataset, the use of LSTM classification models for each class, and the evaluation of detection results against the ground truth. The algorithm's systematic approach involves training the model, detecting malware, and evaluating performance for each class individually. This ensures a comprehensive analysis of malware detection across different classes, leading to a detailed understanding of the algorithm's effectiveness in detecting various types of malware. Overall, the algorithm provides a clear framework for learning-based malware detection, highlighting the importance of preprocessing, model building, training, detection, and evaluation. By breaking down the process for each class, the algorithm offers a methodical and thorough approach to malware detection, contributing to a deeper understanding of the performance and effectiveness of the detection process across different classes of malware.

4.6 Evaluation Method

Given the highly unbalanced nature of our dataset, relying solely on typical accuracy-based performance evaluation would be insufficient to determine the best classifier. To address this, we've implemented an evaluation method that leverages four standard measures in information retrieval, as detailed in [8]. These measures, including total prediction accuracy, average recall, average precision [9], and F1 score, provide a more comprehensive and reliable assessment of the algorithm's performance, particularly in real-world scenarios where dataset imbalance is common.

5. EXPERIMENTAL RESULTS

The suggested framework's experimental findings are shown in this section. The observations are compared with the results of many existing models, like K-Nearest Neighbours (KNN), XGBC, and a baseline CNN model. The empirical study includes creating a malware dataset based on Windows API calls and using it with a supervisor learning approach. The proposed methodology includes an LSTM-based classifier designed to effectively detect every class of malware.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 300)	240000
spatial_dropout1d (SpatialDr	(None, 100, 300)	0
lstm (LSTM)	(None, 100, 32)	42624
lstm_1 (LSTM)	(None, 100, 32)	8320
lstm_2 (LSTM)	(None, 32)	8320
dense (Dense)	(None, 128)	4224
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
out_layer (Dense)	(None, 1)	129

Total params: 369,537
Trainable params: 369,537
Non-trainable params: 0

Figure 4: The layers involved in the LSTM-based deep learning model

As presented in Figure 4, several layers are involved in the proposed deep learning model. 3 LSTM layers follow the embedding layer and spatial dropout layer. Dropouts are appropriately configured to enhance the model's functionality. The model is responsible for analyzing and classifying a given test sample efficiently.

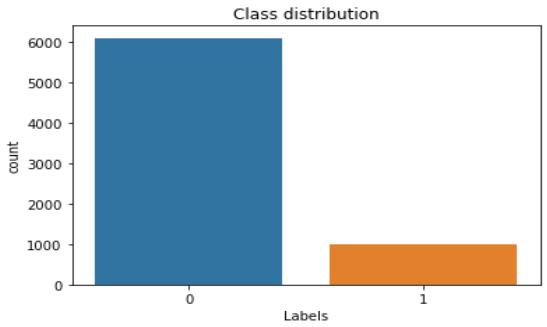

Figure 5: Distribution of classes in the dataset

Figure 5 presents the data distribution dynamics pertaining to both malware and normal instances in the given dataset.

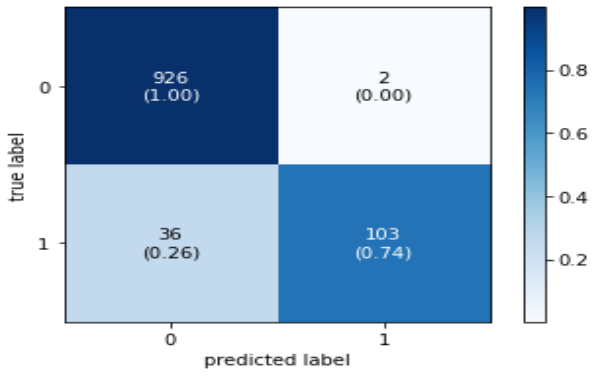

Figure 6: Confusion matrix showing the comparison between predicted labels and ground truth

As seen in Figure 6 the confusion matrix reflects the proposed deep learning framework's performance in classifying malware. The data presented in the confusion matrix are the basis for evaluating the proposed model.

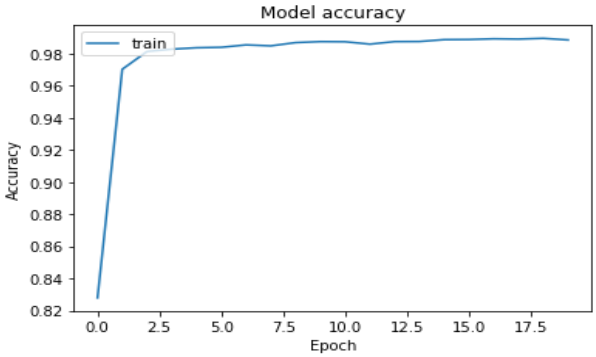

Figure 7: Model accuracy against a number of epochs

In Figure 7, the proposed model's performance is visualized against several epochs. Accuracy is the major metric determining The effectiveness of the suggested deep learning model. Superior performance is indicated by higher accuracy. The model's accuracy steadily rises with the number of epochs. The highest accuracy is evident at the convergence point.

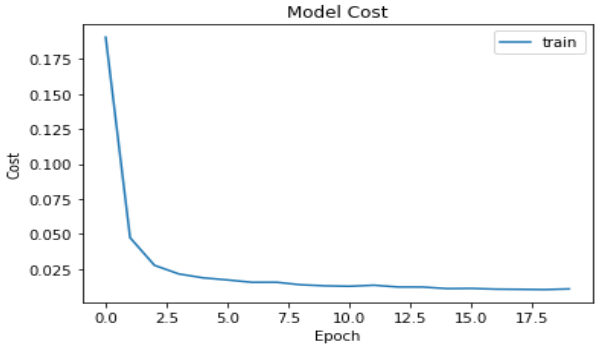

Figure 8: Model loss against a number of epochs

The suggested deep-learning model's model loss dynamics are displayed versus the number of epochs in Figure 8. One measure used to assess the effectiveness of deep learning models is loss value. Improved performance is indicated by a low loss value. The loss value of the deep learning model steadily drops as the number of epochs rises. At the convergence point, the lowest loss value is shown.

Malware Detection Model	Precision	Recall	F1-Score	Accuracy
KNN	93.385	93.936	93.6605	93.6225
XGBC	94.368	94.9248	94.6464	94.608
Baseline CNN	96.334	96.9024	96.6182	96.579
LSTM (Proposed)	98.8012	98.78341	98.7923	98.7923

Table 2: Performance comparison among different models

As presented in Table 2, it is evident that the malware detection performance of different models is provided.

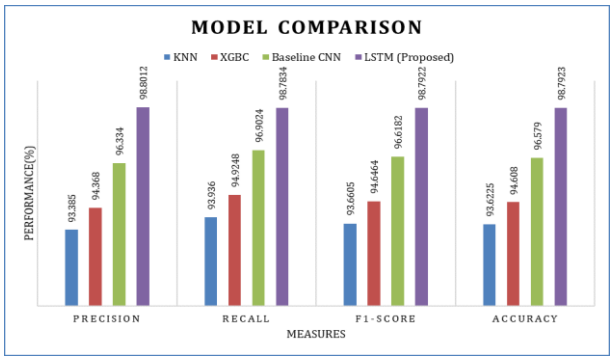

Figure 9: Performance comparison among different malware detection models

The malware detection performance levels of the various models were clearly different, as Figure 9 shows. The findings indicate that the suggested approach based on long short-term memory might outperform the current state

of the art in terms of performance. The reasoning for this is that, in addition to employing the LSTM model effectively, the suggested solution leverages dataset quality. The precision of the KNN model is 93.3850%, XGBC is 94.3680%, the baseline CNN model is 96.3340%, and the proposed LSTM model is 98.8012%. With respect to recall, KNN achieved 93.9360%, XGBC 94.9248%, and baseline CNN model 96.9024%, while the proposed LSTM model could achieve 98.7834% recall. With regard to the F1-score, the KNN model exhibited 93.6605%, the XGBC model 94.6464%, and the baseline CNN model 96.6182%, while the proposed LSTM-based model could achieve 98.7922%. The accuracy exhibited by the KNN Model is 93.6225%, XGBC 94.6080%, and the baseline CNN model 96.5790%, while the proposed LSTM model could achieve 98.7923% accuracy. Comparing the suggested LSTM-based methodology to state-of-the-art techniques, the findings demonstrate that it has the greatest accuracy, at 98.7923%.

6. DISCUSSION

Malware is a problem that has been increasing over many years, with an increase in the number of incidents every year. Traditional approaches, using heuristics-based methods, could detect malware. However, the traditional approaches need to learn from the historical data to gain knowledge incrementally. The advent of AI-based methods has made it feasible to identify known and new malware samples and comprehend instances of malware that are now in circulation. Since malware exhibits malicious behavior by making API calls with the operating system, the proposed methodology is based on an examination of Windows API calls. The proposed method has provision for dataset creation that is suitable for efficient malware detection. It has a solid preprocessing methodology for improving the quality of training data. An LSTM-based deep learning model with several layers is proposed to classify given malware instances. The proposed framework can be used to detect different classes of malware. However, the proposed deep learning-based framework has certain limitations, as discussed in section 5.1.

6.1 Limitations

Modern models are not as precise as the suggested framework for effective virus detection.However, it has some specific limitations. The research is based on historical calls to Windows API. The detection of malware is based on behavioral patterns when calling Windows API. Other behavioral patterns of malware might not be considered in this research. Another significant limitation is that the proposed deep learning framework is trained with a specific dataset. To generalize its functionality, training the model with diversified datasets is important. Before LSTM based classification model it is also possible to analyze data with machine learning algorithms. There is also a need for building hybrid deep learning models to leverage performance in malware detection.

7. CONCLUSION AND FUTURE WORK

We presented a deep learning architecture utilizing Long Short Term Memory (LSTM) for effective virus detection in cybersecurity. Our approach, which we called Learning-Based Malware Detection (LBMD), which exploits the efficient dataset created as part of the framework and uses the LSTM model appropriately to detect malware efficiently. The algorithm provides a clear framework for learning-based malware detection, highlighting the importance of preprocessing, model building, training, detection, and evaluation. By breaking down the process for each class, the algorithm offers a methodical and thorough approach to malware detection, contributing to a deeper understanding of the performance and effectiveness of the detection process across different classes of malware. Our empirical study with the dataset created reveals that the proposed algorithm outperforms many existing malware detection models like KNN, XGBC, and baseline CNN with the highest accuracy of 98.79%. The proposed framework can also be utilized to detect different kinds of malware. The LSTM-based classification model can detect any metamorphic malware, as demonstrated in this paper. In the future, we intend to use a malware image-based approach with deep learning to detect malware efficiently. Another approach to improving the proposed methodology is using sequential data classification models before applying the deep learning method.

REFERENCES

[1] Gautam Karat, Jinesh M. Kannimoola, Namrata Nair, Anu Vazhayil, Sujadevi V G and Prabaharan Poornachandran. (2024). CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection. *Elsevier*. 233, pp.492-503. https://doi.org/10.1016/j.procs.2024.03.239

[2] Santosh K. Smmarwar, Govind P. Gupta and Sanjay Kumar. (2024). Android malware detection and identification frameworks leveraging the machine and deep learning techniques: A compare. *Elsevier*. 14, pp.1-18. https://doi.org/10.1016/j.teler.2024.100130

[3] Ahmed Bensaoud, Jugal Kalita, Mahmoud Bensaoud. (2024). A survey of malware detection using deep learning. *Elsevier*. 16, pp.1-16. https://doi.org/10.1016/j.mlwa.2024.100546

[4] Antonio Galli, Valerio La Gatta, Vincenzo Moscato, Marco Postiglione, and Giancarlo Sperlì. (2024). Explainability in AI-based behavioral malware detection systems. *Elsevier*. 141, pp.1-17. https://doi.org/10.1016/j.cose.2024.103842

[5] Roopa Devi E. M, Naif Almakayeel and E. Laxmi Lydia. (2024). Improved sand cat swarm optimization with deep learning based enhanced malicious activity recognition for cybersecurity. *Elsevier*. 98, pp.187-198. https://doi.org/10.1016/j.aej.2024.04.053

[6] Ahsan Nazir, Jingsha He, Nafei Zhu, Saima Siraj Qureshi, Siraj Uddin Qureshi, Faheem Ullah, Ahsan Wajahat and Muhammad Salman Pathan. (2024). A deep learning-based novel hybrid CNN-LSTM architecture for efficient detection of threats in the IoT ecosystem. *Elsevier*. 15(7), pp.1-21. https://doi.org/10.1016/j.asej.2024.102777

[7] RANDA ALLAFI AND IBRAHIM R. ALZAHRANI. (2024). Enhancing Cybersecurity in the Internet of Things Environment Using Artificial Orca Algorithm and Ensemble Learning Model. *IEEE*. 12, pp.63282 - 63291. http://DOI:10.1109/ACCESS.2024.3390093

[8] Elliot Mbunge, Benhildah Muchemwa, John Batani and Nobuhle Mbuyisa. (2023). A review of deep learning models to detect malware in Android applications. *Elsevier*. 1, pp.1-9. https://doi.org/10.1016/j.csa.2023.100014

[9] GIOVANNI APRUZZESE, PAVEL LASKOV, EDGARDO MONTES DE OCA, WISSAM MALLOULI, LUIS BÚRDALO RAPA, ATHANASIOS VASILEIOS GRAMMATOPOULOS and FABIO DI FRANCO. (2023). The Role of Machine Learning in Cybersecurity. *ACM*. 4(1), pp.1-38. https://doi.org/10.1145/3545574

[10] HAYAM ALAMRO, WAFA MTOUAA, SUMAYH ALJAMEEL, AHMED S. SALAMA, MANAR AHMED HAMZA, AND ALADDIN YAHYA OTHMAN. (2023). Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity. *IEEE*. 11, pp.72509 - 72517. http://DOI:10.1109/ACCESS.2023.3294263

[11] Mumtaz Ahmed, Neda Afreen, Muneeb Ahmed, Mustafa Sameer and Jameel Ahamed. (2023). An inception V3 approach for malware classification using machine learning and transfer learning. *Elsevier*. 4, pp.11-18. https://doi.org/10.1016/j.ijin.2022.11.005

[12] Marwa Keshk, Nickolaos Koroniotis, Nam Pham, Nour Moustafa, Benjamin Turnbull and Albert Y. Zomaya. (2023). An explainable deep learning-enabled intrusion detection framework in IoT networks. *Elsevier*. 639, pp.1-20. https://doi.org/10.1016/j.ins.2023.119000

[13] P. Vijayalakshmi and D. Karthika. (2023). Hybrid dual-channel convolution neural network (DCCNN) with spider monkey optimization (SMO) for cyber security threats detection in internet of things. *Elsevier*. 27, pp.1-12. https://doi.org/10.1016/j.measen.2023.100783

[14] D. Santhadevi and B. Janet. (2023). Stacked deep learning framework for edge-based intelligent threat detection in IoT network. *Springer*. 79, pp.1-34. https://doi.org/10.1007/s11227-023-05153-y

[15] Nan Sun, Ming Ding, Jiaojiao Jiang, Weikang Xu, Xiaoxing Mo, Yonghang Tai, and Jun Zhang. (2023). Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives. *IEEE*. 25(3), pp.1748 - 1774. http://DOI:10.1109/COMST.2023.3273282

[16] Mayra Macas, Chunming Wu and Walter Fuertes. (2022). A survey on deep learning for cybersecurity: Progress, challenges, and opportunities. *Elsevier*. 212, pp.1-40. https://doi.org/10.1016/j.comnet.2022.109032

[17] Rajasekhar Chaganti, Vinayakumar Ravi and Tuan D. Pham. (2022). Deep learning based cross architecture internet of things malware detection and classification. *Elsevier*. 120, pp.1-22. https://doi.org/10.1016/j.cose.2022.102779

[18] MOHAMMED SALEH ALI MUTHANNA, REEM ALKANHEL, AMMAR MUTHANNA, AHSAN RAFIQ, AND WADHAH AHMED MUTHANNA ABDULLAH. (2022). Towards SDN-Enabled, Intelligent Intrusion Detection System for Internet of Things (IoT). *IEEE*. 10, pp.22756 - 22768. http://DOI:10.1109/ACCESS.2022.3153716

[19] Marek Pawlicki, Rafał Kozik and Michał Choras. (2022). A survey on neural networks for (cyber-) security and (cyber-) security of neural networks. *Elsevier*. 500, pp.1075-1087. https://doi.org/10.1016/j.neucom.2022.06.002

[20] NICOLA CAPUANO, GIUSEPPE FENZA, VINCENZO LOIA, AND CLAUDIO STANZIONE. (2022). Explainable Artificial Intelligence in CyberSecurity: A Survey. *IEEE*. 10, pp.93575 - 93600. http://DOI:10.1109/ACCESS.2022.3204171

[21] Jeffrey C. Kimmel; Andrew D. Mcdole; Mahmoud Abdelsalam; Maanak Gupta and Ravi Sandhu; (2021). Recurrent Neural Networks Based Online Behavioural Malware Detection Techniques for Cloud Infrastructure. IEEE Access. http://doi:10.1109/access.2021.3077498

[22] Ikram Ul Haq; Tamim Ahmed Khan and Adnan Akhunzada; (2021). A Dynamic Robust DL-Based Model for Android Malware Detection. IEEE Access. http://doi:10.1109/access.2021.3079370

[23] Jahromi, Amir Namavar; Hashemi, Sattar; Dehghantanha, Ali; Parizi, Reza M. and Choo, Kim-Kwang Raymond (2020). An Enhanced Stacked LSTM Method With No Random Initialization for Malware Threat Hunting in

Safety and Time-Critical Systems. IEEE Transactions on Emerging Topics in Computational Intelligence, 1–11. http://doi:10.1109/TETCI.2019.2910243

[24] Jahromi, Amir Namavar; Hashemi, Sattar; Dehghantanha, Ali; Parizi, Reza M. and Choo, Kim-Kwang Raymond (2020). An Enhanced Stacked LSTM Method With No Random Initialization for Malware Threat Hunting in Safety and Time-Critical Systems. IEEE Transactions on Emerging Topics in Computational Intelligence, 1–11. http://doi:10.1109/TETCI.2019.2910243

[25] Bibi, Iram; Akhunzada, Adnan; Malik, Jahanzaib; Iqbal, Javed; Mussaddiq, Arslan and Kim, Sungwon (2020). A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware. IEEE Access, 8, 129600–129612. http://doi:10.1109/ACCESS.2020.3009819

[26] Sirajuddin Qureshi; Jingsha He; Saima Tunio; Nafei Zhu; Faheem Akhtar; Faheem Ullah; Ahsan Nazir andAhsan Wajahat; (2021). A Hybrid DL-Based Detection Mechanism for Cyber Threats in Secure Networks . IEEE Access. http://doi:10.1109/access.2021.3081069

[27] VINAYAKUMAR R, MAMOUN ALAZAB, SOMAN KP, PRABAHARAN POORNACHANDRAN, and SITALAKSHMI VENKATRAMAN. (2019). Robust Intelligent Malware Detection Using Deep Learning. *IEEE*. 7, pp.46717 - 46738. http://DOI:10.1109/ACCESS.2019.2906934

[28] Eva Rodriguez; Beatriz Otero; Norma Gutierrez and Ramon Canal; (2021). A Survey of Deep Learning Techniques for Cybersecurity in Mobile Networks . IEEE Communications Surveys & Tutorials. http://doi:10.1109/COMST.2021.3086296

[29] Junyang Qiu; Jun Zhang; Wei Luo; Lei Pan; Surya Nepal and Yang Xiang; (2021). A Survey of Android Malware Detection with Deep Neural Models . ACM Computing Surveys. http://doi:10.1145/3417978

[30] Shukla, Sanket; Kolhe, Gaurav; PD, Sai Manoj and Rafatirad, Setareh (2019). 18th IEEE International Conference On Machine Learning And Applications (ICMLA) - RNN-Based Classifier to Detect Stealthy Malware using Localized Features and Complex Symbolic Sequence. 406–409. http://doi:10.1109/icmla.2019.00076

[31] Fang, Xing; Xu, Maochao; Xu, Shouhuai and Zhao, Peng (2019). A deep learning framework for predicting cyber attacks rates. EURASIP Journal on Information Security, 2019(1), 5–. http://doi:10.1186/s13635-019-0090-6

[32] Amin, Muhammad; Tanveer, Tamleek Ali; Tehseen, Mohammad; Khan, Murad; Khan, Fakhri Alam and Anwar, Sajid (2019). Static malware detection and attribution in android byte-code through an end-to-end deep system. Future Generation Computer Systems, S0167739X19308490–. http://doi:10.1016/j.future.2019.07.070

[33] Onur Barut; Matthew Grohotolski; Connor DiLeo; Yan Luo;Peilong Li and Tong Zhang; (2020). Machine Learning Based Malware Detection on Encrypted Traffic: A Comprehensive Performance Study . 7th International Conference on Networking, Systems and Security. http://doi:10.1145/3428363.3428365

[34] Li, Yi; Xiong, Kaiqi; Chin, Tommy and Hu, Chengbin (2019). A Machine Learning Framework for Domain Generation Algorithm (DGA)-Based Malware Detection. IEEE Access, 1–1. http://doi:10.1109/ACCESS.2019.2891588

[35] Mani, Ganapathy; Pasumarti, Vikram; Bhargava, Bharat; Vora, Faisal Tariq; MacDonald, James; King, Justin and Kobes, Jason (2020). IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS) - DeCrypto Pro: Deep Learning Based Cryptomining Malware Detection Using Performance Counters. 109–118. http://doi:10.1109/ACSOS49614.2020.00032

[36] Mahdavifar, Samaneh and Ghorbani, Ali A. (2019). Application of Deep Learning to Cybersecurity: A Survey. Neurocomputing, S0925231219302954–. http://doi:10.1016/j.neucom.2019.02.056

[37] Samy, Ahmed; Yu, Haining and Zhang, Hongli (2020). Fog-Based Attack Detection Framework for Internet of Things Using Deep Learning. IEEE Access, 8, 74571–74585. http://doi:10.1109/ACCESS.2020.2988854

[38] Yin, Jiao; Tang, MingJian; Cao, Jinli and Wang, Hua (2020). Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. Knowledge-Based Systems, 210, 106529–. http://doi:10.1016/j.knosys.2020.106529

[39] Hasan, Tooba; Adnan, Akhunzada; Giannetsos, Thanassis and Malik, Jahanzaib (2020). 6th IEEE Conference on Network Softwarization (NetSoft) - Orchestrating SDN Control Plane towards Enhanced IoT Security. 457–464. http://doi:10.1109/NetSoft48620.2020.9165424

[40] S, Akarsh; K, Simran; Poornachandran, Prabaharan; Menon, Vijay Krishna and P, Soman K (2019). 5th International Conference on Advanced Computing & Communication Systems (ICACCS) - Deep Learning Framework and Visualization for Malware Classification. 1059–1063. http://doi:10.1109/ICACCS.2019.8728471