Multichannel Based IoT Malware Detection System Using System Calls and Opcode Sequences

Shobana Manoharan
Department of Computer Science and Engineering, Rajalakshmi Engineering College, India
divyashobana.m@gmail.com

Poonkuzhali Sugumaran
Department of Computer Science and Engineering, Rajalakshmi Engineering College, India
poonkuzhali.s@rajalakshmi.edu.in

Kishore Kumar
AI Engineer, National Institute of Fashion Technology, India
kishore.kumar@nift.ac.in

Abstract: *The rapid development in the field of the Internet of things gives rise to many malicious attacks, since it holds many smart objects whose lack of an efficient security framework. These kinds of security issues bring the entire halt-down situation to all smart objects that are connected to the network. In this work, multichannel Convolutional Neural Network (CNN) is proposed whereas each channel's CNN works on each type of input parameter. This model has two channels connected in a parallel manner, with one CNN taking an opcode sequence as input and the other CNN running with system calls. These extracted system calls and opcode sequences of elf files were discriminated against using two more deep learning algorithms along with multichannel CNN, namely Recurrent Neural Network (RNN) and CNN, and a few recent existing solutions. The performance analysis of the aforementioned algorithms has been carried out and evaluated using accuracy, precision, recall, F1-measure, and time. The experimental results show that multichannel CNN outperforms the remaining considered techniques by achieving a high accuracy of 99.8% for classifying malicious samples from benign ones. The real-time Internet of Things (IoT) malware samples were collected from the IoT honeyPot (IOTPOT), which emulates different CPU architectures of IoT devices.*

Keywords: *System calls, IoT malwares, fog computing, RNN, CNN, multichannel CNN.*

Received November 27, 2020; accepted July 29, 2021
https://doi.org/10.34028/iajit/19/2/13

1. Introduction

In recent decades, the term "Internet of things" has been involved in the development of diverse technologies like military, medical, automobile, industrial, and even domestic purposes like smart homes. This scenario increases the number of smart objects over the network drastically [10]. Hence, this sudden demand for smart objects at a high rate leaves vendors to design smart objects without any efficient security framework. Due to the lack of a heavyweight security framework in the IoT environment, these kinds of smart devices are easily targeted by attackers [24]. Many recent malwares, such as Mirai [3], are capable of launching high-speed floods of DDoS attacks. In this regard, many researchers have started to focus on this security issue for IoT with a few traditional as well as emerging technologies like Software-Defined Networking (SDN), fog [15], Game theory, Cryptography [9], behaviour profiling [8] and Artificial intelligence [1, 12, 18]. Since IoT is said to be a resource constrained and heterogeneous platform, there are some challenges that exist while designing a security framework for IoT platforms. First, the designed solution has to be a very light-weight algorithm without consuming more power, time, or processing speed. Secondly, the design framework should be very flexible and reliable for all sorts of diverse technologies connected to the IoT domain [17]. In general, the existing solutions for malware detection can be broadly classified as signature-based methods [2], behavior/anomaly-based detection [4], and sandbox/honeypot-based analysis [20]. Among these techniques, the signature-based approach [2] achieves higher accuracy in detection rate, but it fails to detect the new kinds of malicious attacks. It is clear that, for the current scenario, the signature-based method is not suitable to detect malware in the IoT environment. In this regard, researchers have focused much work towards anomaly/behavior-based methodology, and this approach is coming into reality by incorporating powerful ML [4, 5] and Deep Learning (DL) algorithms [9]. When comparing the efficiency of machine learning [4, 5] with deep learning techniques, the latter achieves better performance, especially in the field of malware/intrusion detection [18]. The efficacy of any deep learning architecture is always determined by the input features and its optimized hyper parameters. The selection of input attributes in such a way that they should be highly capable of reflecting the behaviour of the incoming malware file. By this way, the selection of the input features to disassemble malware binaries can be broadly classified into static [9] and dynamic malware analysis [14]. Static malware analysis [9] is usually carried out by generating the behaviour of the malware file without unpacking the

binary malware executable. An example of static analysis [9] is the cryptographic hash or unique signature of the malware file to exhibit the property of the particular file. Unlike static analysis, dynamic analysis [14] tends to execute the suspicious samples in an isolated environment to trace the malicious activities or behaviour of the new variants of malware. In order to prevent the host system from damage launched by malware samples. Hence, it can be concluded that dynamic analysis is always efficient for detecting the occurrence of a new variant of malware files when compared to static analysis [8]. Network traffic, opcode sequences [1], Printable Strings Information (PSI) graph, system calls [19, 24], binary image, and Application Programmable Interface (API) calls are the most common input attributes considered for dynamic malware analysis.

Out of these attributes, system calls [19, 23] and opcode sequences [1] were considered as effective attributes to extract the behaviour of malicious and benign samples, so these attributes were used in this work in the multichannel deep learning architecture. However, each feature has its own merits and demerits in detecting the new malware, so here we used two aforementioned significant features in a parallel manner to improve detection rate in either way. Two DL algorithms have been incorporated for malware detection to achieve high performance [4, 11, 19, 22], but this approach has a high computational workload. So, in this proposed work, this issue has been addressed by designing two DL in a parallel manner by overlapping a few of the common neural layers (preprocessing layer). The main motivation for this work is to provide a reliable and flexible malware detection system for IoT-based environments or smart/IoT objects. The proposed work is based on the deployment of a malware detection system at the fog layer, which is present in the IoT architecture. The proposed solution should be able to detect the unknown variant of a new malware sample, unlike the signature-based malware detection technique [2].

The overall contribution of this work is summarized as given below is

1. To generate the system call and opcode sequences of both normal and IoT malicious files
2. To preprocess the extracted sequences using vectorization and word embedding method
3. To do comparative analysis of above mentioned two feature processing techniques
4. To classify the generated input attributes as malware and benign separately using deep learning techniques
5. To do performance analysis of proposed multichannel Convolutional Neural Network (CNN) against Multichannel Long Short-term Memory (LSTM), CNN, Recurrent Neural Network (RNN) and machine learning techniques using various quality metrics.

The organization of the paper can be given as section 2 explains the literature survey relevant to the proposed work and it is discussed in a detailed manner. Section 3 talks about the steps involved in the proposed model. Section 4 explores the results of the proposed model along with its experimental setup utilized for this work. Finally, section 5 ends with the conclusion and future enhancements.

2. Motivation of the Work

The main motive behind the work is the handling of a greater number of features at the same time using a single model. This type of model can be achieved by implementing a multichannel architecture imposed on top of deep learning algorithms. The restriction of usage of a single feature as an input attribute of any deep learning or machine learning technique results in a well-trained model capable of detecting malware whose behavior may be prone to that particular chosen feature. The different behavior of malware invariants is analyzed using different features of malware samples. To achieve this, with the minimum computational workload, a multichannel architecture has been chosen to handle each input attribute in each channel. If the format of the input attributes considered for multichannel purposes belongs to the same data type, then the data preprocessing step of these inputs can be merged instead of a separate block for each channel. For the IoT environment, the distributive nature of the security solution has been chosen to provide reliable services on-time rather than a centralized approach. This kind of approach can be incorporated by implementing a malware engine in fog nodes rather than on the IoT devices themselves.

3. Related Works

This section describes the existing approaches which are specifically designed to detect malware using system calls as well as opcode sequences.

Kolosnjaji *et al*. [16] proposed a model that combines two major deep learning architectures, namely CNN and RNN, into a single deep neural network. This approach is tested against traditional machine learning techniques such as Support Vector Machine as well as the Hidden Markov Model to prove its efficiency. The combined model works well for system call sequences as an input attribute. Hou et al [12] designed a deep learning security model for android malware by using system call sequences. In this work, the author has generated a system call graph from the extracted system calls of the android malware samples. These graphs are then passed as input attributes to the stacked auto encoders for training. Xiao *et al*. [23] implemented a deep learning model for android malware using system call sequences. The

author deployed two LSTM networks, one for training normal system call sequences and another one for training malicious system call sequences. Mishra *et al*. [19] presented a deep learning-based security model for cloud platforms. The author has utilised system calls for malware analysis and implemented two stage classification using CNN and Bi-directional LSTM. Kim *et al*. [18] proposed a deep learning model for android malware detection using similarity calculation. The author has extracted several kinds of features from the malware samples. Out of those features, the most prominent feature will be selected based on similarity calculation and those features are further classified using a deep neural network. Khater *et al*. [15] proposed a security framework for the IoT environment using fog computing. The author utilised two recent system call datasets such as Australian Defence Force Academy Linux Dataset (ADFA-LD) and ADFA-WD, then the designed model was implemented on the Raspberry Pi as fog nodes. This model uses single-layer perceptrons to classify malicious system calls from normal ones.

Apart from the above discussed existing techniques, some more existing solutions are as follows: which relies on Linux malware detection techniques based on the system call approach.

An *et al*. [4] proposed a security framework for home routers in order to protect them from Distributed Denial of Service (DDoS) attacks. Here, the author utilised three machine learning techniques for classification, namely Principal Component Analysis (PCA), one-class SVM, and Nave Bayes. Breitenbacher *et al*. [6] designed HADES-IoT, a security framework for IoT devices. It works on the basis of whitelisting legitimate IoT devices based on the system calls collected from real-time IoT malware. The author proves that this work is a lightweight model for IoT devices because it can be deployed on any type of IoT device. Abbas and Srikanthan [2] presented a signature-based malware detection method specifically for IoT devices. The aforementioned signatures were retrieved from the system calls of the Linux based malware. These signatures are combined in such a way that malware that belongs to the same family will have a single signature. An *et al* [5] suggested a malware detection system for the Amazon Alexa Echo using system calls from IoT malware. The extracted system calls were classified as benign and malicious using one class support vector machine. Furthermore, the performance of the classifier is evaluated and analysed using the Cumulative Sum control chart (CUSUM) test. Haddad *et al*. [11] proposed a malware detection system for the IoT platform using RNN-LSTM by extracting the opcode sequences of Advanced RISC Machine (ARM)-based IoT malware. The author incorporates three different LSTM models for classification purposes. Darabian *et al*. [7] suggested a security model capture, especially for polymorphic malware, exists in the IoT domain. Here the author has made use of the maximal opcode sequence as the input parameter to distinguish between malicious and benign samples, and the sequence classification was carried out using K nearest neighbors, support vector machines, multilayer perceptron, AdaBoost, decision tree, and random forest classifier. Azmoodeh *et al*. [1] proposed malware detection for Internet-based Battle Things (IoBT) using opcode sequences. The author transforms extracted sequences into vector space and then they are further classified into benign and malicious sequences correspondingly using deep eigenspace learning. Additionally, some of the recent work in IoT has been analysed and compared with the proposed methodology. Zielonka *et al*. [25] analyse the trend existing in the growth of smart home technology. This analysis extends in terms of various parameters such as energy management, security in databases, and quality of life. Iwendi *et al*. [13] proposed a watermarking-based approach to counter cyberattacks. It is done by imposing two watermarks on a single piece of software, and this technique is so called "keysplitwatermark".

At the end of this survey, it reveals that most of the existing model is not evaluated using real time IoT malware samples. Moreover, RNN is not used much in the existing work to detect IoT malware. Since RNN is well known for handling natural language processing issues, it is assumed to be suitable for classifying malicious system call sequences from normal system calls. In our previous work [21], RNN was proved to be efficient in detecting the system calls of malicious files residing on IoT devices and that model was further improved and compared with equivalent deep learning and machine learning techniques. From a security point of view, many existing security solutions for smart homes are based on Blockchain technology [25]. However, blockchain is vulnerable to a few major security attacks, including the eclipse attack, the consensus mechanism, and code vulnerabilities. In watermark-based security solutions [13], the watermark present in the software can be removed by modifying the carrier data (software) like compression or resizing. So, this approach cannot be applied to protecting IoT devices from cyberattacks. But deep learning-based solutions are free from these kinds of attacks. Jon *et al*. [14] suggested a cloud-based malware detection framework using CNN. The implementation of a malware engine on a cloud platform connected to IoT devices increases the latency and communication bandwidth. These above highlighted issues discussed in each existing solution strengthen the ideology of the proposed work. Since the cloud node has been replaced by a fog node, which is nearer to many IoT devices in the proposed solution, this approach has proven more effective in providing the services. Moreover, in some of the existing systems, the dynamic features have been converted to

images and they are used as inputs. The storage space of an image is always higher than the other datatypes like strings or numeric, hence those approaches are not suitable for the IoT domain. Hence, in this approach, rather than using a single feature, two of the most significant features, like system calls and opcode sequence, were used in parallel architecture. This style of implementation enhances the detection rate of new variants of malware which pose different malicious behaviour patterns.

4. Proposed Methodology

This section explains the layers of the system architecture and its operation in a detailed manner. Figure1 illustrates the overview of the communication between the fog nodes and the IoT devices which are connected to it. In Figure 2, the deep learning-based security model which is installed in the fog node is demonstrated in a layered manner.

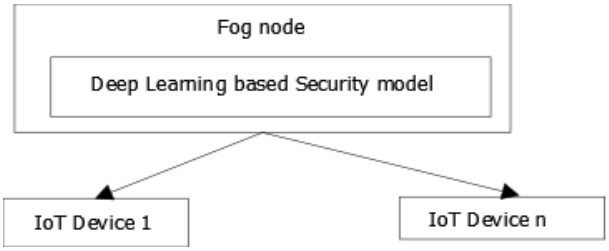


Figure 1. Overview of system architecture.

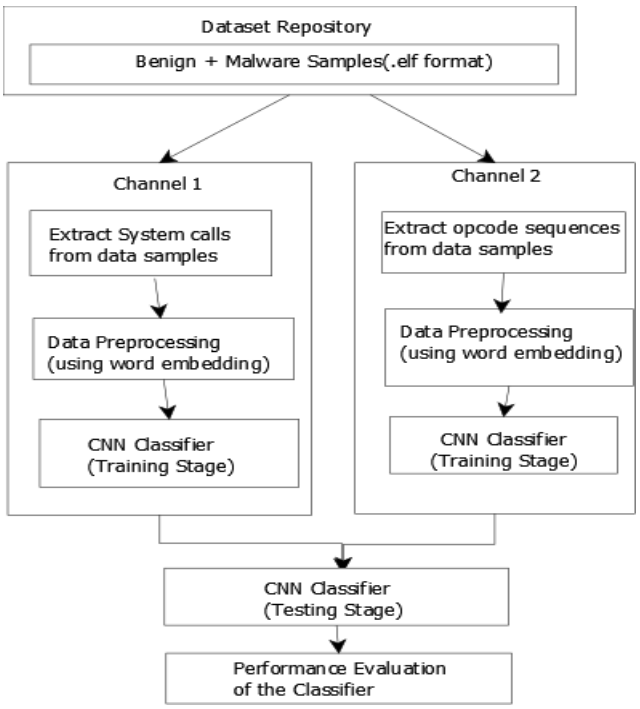


Figure 2. Deep learning based security model.

4.1. Dataset Generation Layer

This dataset comprises different types of system calls extracted from both normal and malware samples residing on IoT devices. Since the majority of the IoT devices are built using the Linux operating system, the benign samples were collected from system files residing in the path/bin,/usr/bin, and/sbin folders of the Ubuntu operating system. Meanwhile, the real-time IoT malware samples were extracted from Internet of things Honeypot (IoTPoT) [21]. IoTPoT is a honeypot specially designed to capture telnet-based attacks on IoT devices. This honeypot emulated nine different Central Processing Unit (CPU) architectures such as ARM, Microprocessor without Interlocked Pipelined Stages (MIPS), Power PC (PPC), Little endian MIPS (MIPSEL), Scalable Processor Architecture (SPARC), MIPS64, SH4 and X86. Therefore, tracing system calls from samples belonging to diverse architectures gives enough knowledge of the designed model to detect unknown incoming malware.

Table 1. System call refinement.

Type	Number of Samples	
	Training	Testing
Malware	3072	1317
Benign	2800	1200

Table 2. Sample distribution for model.

System call	Refined System call
execve("./malware_10", ["./malware_10"], [/* 23 vars */])=0	Execve
uname({sysname="Linux", nodename="264egular-VirtualBox", ...}) = 0	Uname
brk(0x634f4240)=0x634f4240	Brk
arch_prctl(ARCH_SET_FS, 0x634f3900)= 0	arch_prctl
set_tid_address(0x634f3bd0)=6658	set_tid_address
set_robust_list(0x634f3be0, 24)=0	set_robust_list

These normal and benign files were executed using preinstalled software, the so-called strace tool in Ubuntu, which is installed in an isolated environment. In the same way, the objdump command is used to extract opcode sequences from the elf samples. After executing each file, a set of system calls and opcode sequences were returned along with their parameters and return value. In the process of system call refinement, the parameters and their return value were truncated and it is shown in Table 1. Likewise, the opcode sequences are also processed in the same manner. The sample distribution for both system calls and opcodes is illustrated in Table 2.

4.2. Data Preprocessing Layer

The data preprocessing layer include two kinds of approaches such as:

1) N-gram and vectorization.
2) Word embedding and it is explained below.

4.2.1. N-Gram and Vectorization

The N-gram technique is widely applied for text processing. Since system calls are in textual format, the N-gram technique is used in this stage. In this work, an n-gram is used to extract an efficient number of features to predict the next occurrence of a word. Another technique is vectorization, which is used to

allocate weight to each word based on its occurrence. In this work, Term Frequency-Inverse Document Frequency (TF-IDF) is applied, which is used to allocate more weight to the high occurrence of a word in a document whereas lower weight is allocated to the particular low occurrence word. Here the value of 'n' is used in the n-gram technique varies from 1 to 5. The weight of the frequency used in the TF-IDF is 10. After performing these two transformations, the inputs are fed into the neural network (CNN and RNN) in the form of a sparse matrix.

4.3. Word Embedding

Word embedding encodes each word as a feature vector, so it conquers more information from very few text sequences. Tokenization is performed as the first step of word embedding to remove unwanted words or special characters to improve the performance of the model and here words are considered as tokens by mapping each word to an integer. Since each sentence has a different list of words, the pad-sequence() function is used to equate the length of all the vector sizes. Using an embedding layer of keras in Python, these calculated integers are mapped to dense vectors. The parameters used in the embedding layers are input-dim (size of the vocabulary), output-dim (size of the dense vector), and input-length (length of the sequence).

4.4. Classification Layer

In the classification layer, four deep learning architectures have been used, viz., Multichannel CNN, Multichannel LSTM, CNN, and RNN. The purpose of classification is to distinguish the processed system call sequence as either malicious or benign samples respectively. Usually, the deep learning architecture requires a high computational workload and a large number of training samples. Since this work is specially designed for IoT devices, which are said to be resource-constrained devices, a lightweight deep learning architecture is used here with a minimum number of hidden layers which is trained with a considerably small amount of dataset sample.

4.5. Fog Node and IoT Devices

The term fog is meant to be replaced by cloud. In IoT applications, the cloud component is used to provide required services to the edge devices (IoT devices). To reduce the communication latency, fog is introduced which acts as an intermediate between cloud and edge devices. So, in this work, in order to reduce the computational workload for IoT devices, the main deep learning security framework is deployed in a fog node. In an IoT device, a lightweight scanning agent is installed which is used to detect incoming files as malware. If it is completely normal, then no action is performed. In the other case, if any suspicious file is detected, then it will be directed to the fog node for further scanning using deep learning techniques. If any new malware is detected inside the fog node, then the particular rule will be updated in the scanning agent which is installed on the IoT device.

5. Algorithm

This section explains about the algorithm used for the classification task in a detailed manner.

5.1. Recurrent Neural Network

A recurrent neural network, as the name implies, is a type of neural network that uses the previous output as a present input in a recursive manner. This nature of RNN is made suitable for the problem of prediction. The most prominent feature of RNN is the hidden state, which helps to store the information for each sequence. The values for the parameters that predict the sequence have the same value for performing the same task at all the hidden layers.

5.2. Convolutional Neural Network

CNN is one of the deep learning techniques that is quite familiar for dealing with image analytics. Most recently, CNN has been used extensively for text processing and sentiment analysis. It is also known for the regularized type of multilayer perceptron that prevents overfitting, and this model resembles the biological neurons of the human brain. Obviously, like other deep learning techniques, CNN also has one input and output layer along with multiple hidden layers. The hidden layer in the CNN is connected based on the operation convolve, otherwise known as the dot operation.

5.3. Multichannel CNN

Multichannel CNN is a special variant of a convolutional neural network consists of more than one convolutional neural network connected in a parallel manner. The architecture of this model has two CNN connected immediately after their flatten layer using the concatenate layer. In this model, the first channel CNN is based on the system call sequence as an input attribute whereas the second CNN works on the opcode sequence. This model discriminates the malicious sample using two variants of input features that increases the probability of detection rate. There is a chance of predicting the malicious sample based on any one of two different input features (system calls and opcode). The input parameters of these attributes computed for each layer are illustrated in Table 3.

Table 3. Parameters involved in multichannel CNN model.

Layer	Output Shape	Parameters
Input_19	(None,50)	0
Input_20	(None,50)	0
Embedding_19	(None,50,200)	11723800
Embedding_20	(None,50,200)	11723800
Conv1d_19	(None,48,32)	19232
Conv1d_20	(None,48,32)	19232
Dropout_19	(None,48,32)	0
Dropout_20	(None,48,32)	0
Max_pooling1d_19	(None,24,32)	0
Max_pooling1d_20	(None,24,32)	0
Flatten_19	(None,768)	0
Flatten_20	(None,768)	0
Concatenate_10	(None,1536)	0
Dense_19	(None,10)	15370
Dense_20	(None,1)	15370

5.4. MultiChannel LSTM

Like in multichannel CNN, here two LSTM models are connected in a parallel way, and these two connected models work on the basis of two different input features separately. LSTM is a special form of RNN and it is designed to overcome the problem of long-term dependencies. Recently, it has been seeking attention towards text processing for better performance. Since the two input attributes considered in this work are text sequence, the LSTM technique is used here for comparative study. The input parameters of these attributes computed for each layer are illustrated in Table 4. In Table 5, the values used for the three deep learning techniques have been explored in an elaborate manner.

Table 4. Parameters involved in multichannel LSTM model.

Layer	Output Shape	Parameters
Input_19	(None,50)	0
Input_20	(None,50)	0
Embedding_20	(None,50,250)	14654750
Embedding_21	(None,50,250)	14654750
Lstm_20	(None,64)	80640
Lstm_21	(None,64)	80640
Concatenate_10	(None,128)	0
Dense_20	(None,10)	1290
Dense_21	(None,1)	1290

Table 5. Hyperparameters values for deep learning model.

HyperParameters	Values	
	RNN, CNN	MultiChannel-CNN and Multichannel LSTM
Epochs	3	
Hidden layers	1	Channel1 -1 Channel2 -1
Activation function	output layer-sigmoid function Hidden layer-Relu function	
Neurons	4 neuron-input layer 4 -1st hidden layer 1-output layer	10-hidden Layer 1-output layer Filter Size=32
Optimizer	Adam	
Metrics	Accuracy, Val-Accuracy ,Loss, Val-Loss	Accuracy, Loss
Loss Function	Binary cross entropy	

6. Results and Discussion

The generated system calls were extracted using the preinstalled tool called strace and another tool to extract opcode is objdump in Ubuntu 16.04 installed on the oracle Virtual Machine (VM) virtualbox version 6.0.14 whereas the host computer is windows 8.1. The k-fold cross validation technique is implemented for multichannel CNN alone to prove its efficiency for a minimal number of data samples and here the value of 'k' is 10.

The deep learning techniques such as Multichannel LSTM, Multichannel CNN, Recurrent neural network, and Convolutional neural network were implemented on a Core i3 Laptop with a 2.30 GHz CPU and 4 GB RAM using the Python version 3.7 software environment. The performance analysis of the designed model is measured using the following performance monitors:

- True Positive (TP) shows that the malicious system calls sequences are correctly predicted as malware.
- True Negative (TN) shows that normal system calls sequences are correctly predicted as normal.
- False Positive (FP) shows that normal system calls sequences is wrongly detected as an attack.
- False Negative (FN) shows that the malicious system calls sequences is wrongly detected as normal.

6.1. Accuracy

Accuracy can be defined as the ratio between the number of correctly predicted sequences and the total number of samples, and it is calculated using Equation (1).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

6.2. Precision

Precision can be termed as the ability of the classifier to correctly label malware samples as attacks. Equation (2) is used to calculate the precision of the classifier.

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

6.3. Recall or Detection Rate

"Recall or "detection rate" can be defined as the number of correctly detected malicious samples. Equation (3) is used to calculate the recall of the classifier.

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

6.4. F-Measure

F-measure can be defined as the weighted harmonic mean of precision and recall. Equation (4) is used to calculate the F-measure of the classifier.

$$F - measure = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (4)$$

In Table 6, the comparative and performance analysis has been demonstrated for multichannel LSTM, multichannel CNN, CNN and RNN using the quality metrics. There are two text processing techniques that were employed for CNN and RNN. By comparing those results, it can be inferred that in the case of the system call dataset, both CNN and RNN achieve high performance for n-gram than word embedding, but their training and prediction time is quite higher than the latter. For the opcode sequence dataset, CNN performs well for the word embedding technique in all aspects, but for RNN, processing time is higher for n-gram than for word embedding. On the whole, it can be concluded that the word embedding technique suits faster processing better than the n-gram. While analysing the results obtained from four deep learning and four machine learning techniques, the proposed method, Multichannel CNN and Multichannel LSTM, outperforms the other techniques which are frequently used for malware detection systems. Even though the multichannel architecture works for two different sets of datasets, its prediction and training time is not so much higher than the remaining techniques. Since this model is applied to tiny IoT devices, both time and classification metrics should also be preferred. Among two multichannel architectures, multichannel CNN outperforms multichannel LSTM in terms of accuracy, recall, F1-measure, and time. The Figures 3 and 6 shows that the precision-recall curve of the multichannel CNN and Multichannel LSTM for 10-fold cross validation respectively. Figure 4 shows the area under the curve that achieves the highest precision as well as recall value and it reaches the mean value of AUCPR of approximately 0.994, where for Multichannel LSTM the mean value of AUCPR reaches only 0.990. Figures 4 and 5 show the receiver operating curve for Multichannel CNN and Multichannel LSTM, respectively, and this curve plots between true positive rate and false positive rate. The value of the RoC for all the 10-fold values lies between 0.99 and 1.00 and its mean value is around 0.99, as shown in Figure 4. Likewise, in Figure 5, the ROC curve for multichannel LSTM, the values lie between 0.998 and 1.00 and its mean value is 0.99 appropriately. Figure 7-a) and 7-b) and Figure 8-a) and 8-b) show the accuracy vs. validation accuracy and loss vs. validation loss plots for the system call dataset and opcode dataset, respectively. Figures 9-a) and 9-b) and 10-a) and 9-b) show the RNN used for system calls and the opcode dataset, respectively.

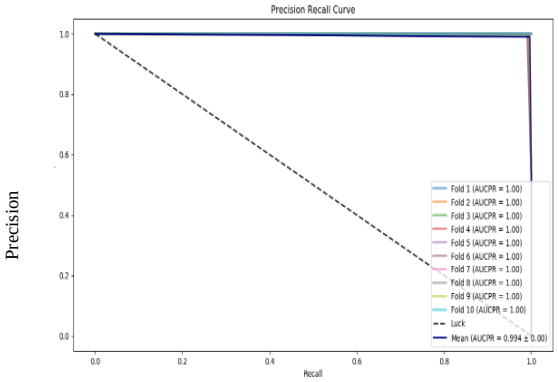


Figure 3. Precision-Recall Curve for Multichannel CNN.

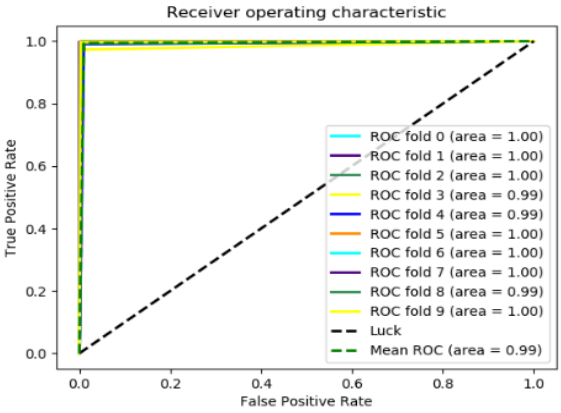


Figure 4. ROC Curve for 10-fold Multichannel CNN.

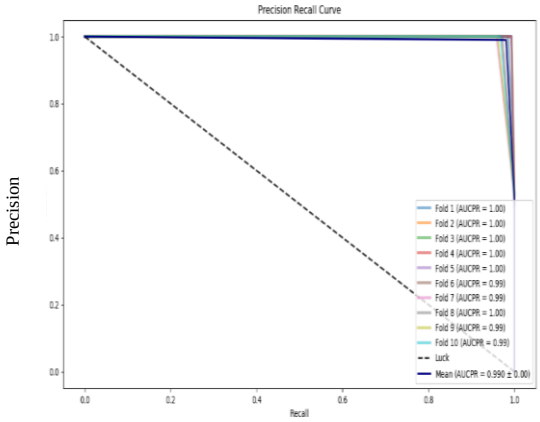


Figure 5. Precision-recall curve for multichannel LSTM.

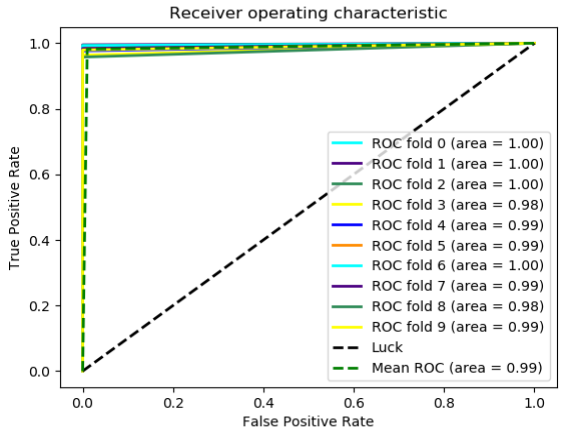
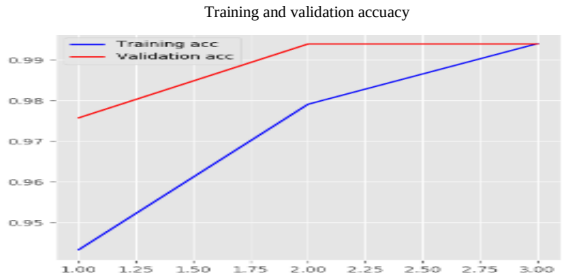
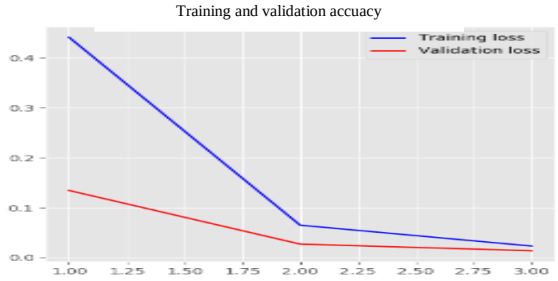


Figure 6. RoC curve for 10-fold multichannel LSTM.

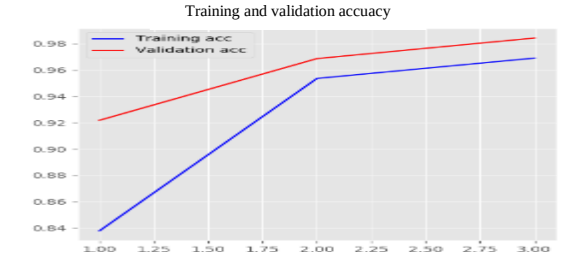


a) Accuracy for training and testing phase.

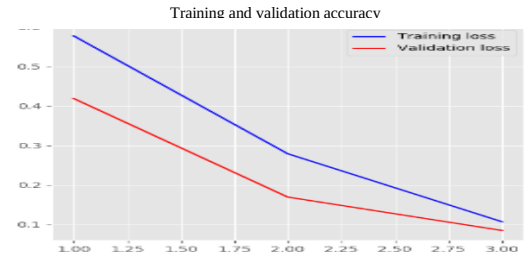


b) Loss for training and testing phase.

Figure 7. CNN for opcode sequence (word embedding).



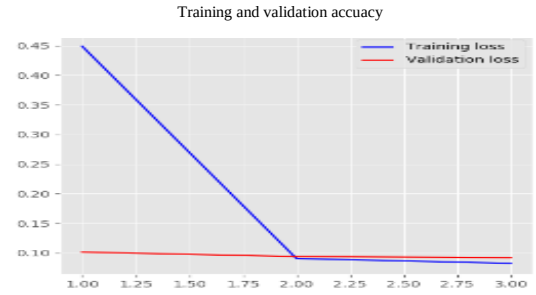
a) Accuracy during training and testing phase.



b) Loss during training and testing phase.

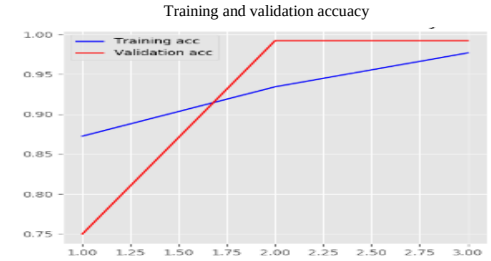
Figure 8. CNN for system call sequence (Word Embedding).

a) Plot between training and testing accuracy.

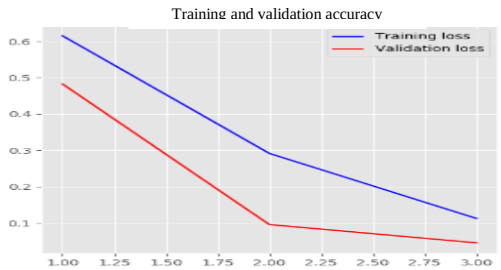


b) Plot between training and testing Loss.

Figure 9. RNN for opcode sequence (word embedding).



a) Plot between training and validation accuracy.



b) Plot between training and validation loss.

Figure 10. RNN for system call sequence (word embedding).

Table 6. Comparative analysis for multichannel CNN, CNN and RNN.

Attribute used	Algorithm used	Text processing techniques	Accuracy(%)	Precision	Recall	F1-measure	Training Time(sec)	Prediction time(sec)
System Call	CNN	n-gram	99.211	0.985	0.985	0.992	54.833	0.208
		Word embedding	98.437	0.970	0.970	0.984	4.96	0.08
	RNN	n-gram	99.214	0.985	0.985	0.992	19.654	0.131
		Word embedding	99.218	0.984	0.984	0.992	5.124	0.574
	SVM	n-gram	99.218	1.000	0.984	0.992	4.022	1.723
	NB	n-gram	98.437	1.000	0.969	0.984	0.655	0.328
	Random Forest	n-gram	98.437	0.970	1.000	0.984	0.558	0.066
	K-nn	n-gram	98.437	0.984	0.984	0.984	1.23	36.04
opcode	CNN	n-gram	99.393	0.989	0.989	0.994	3428.09	39.097
		Word embedding	99.393	0.989	0.989	0.994	156.239	0.113
	RNN	n-gram	99.393	0.989	0.989	0.994	223.119	0.213
		Word embedding	97.575	0.958	0.958	0.978	160.963	0.6
	SVM	n-gram	97.118	0.984	0.984	0.992	5.022	2.000
	NB	n-gram	97.532	0.943	0.969	0.984	2.751	2.228
	Random Forest	n-gram	98.537	0.954	0.969	0.984	0.558	2.134
	K-nn	n-gram	98.537	0.979	0.984	0.984	1.238	48.04
Both	Multichannel CNN	Word Embedding	99.843	0.998	0.998	0.998	41.326	5.7716
	Multichannel LSTM		98.927	1.000	**0.979**	0989	42.664	5.8923

Table 7. Comparative analysis of existing method with proposed methodology.

Author	Dataset used	Attributes used	Techniques and its architecture used	Performance metrics
Kolosnjaji *et al*. [16]	Virus share & Matrieve	System calls	2-layer CNN,RNN	Accuracy-89.4% Precision-0.856 Recall-0.894
Hou *et al*. [12]	Comodo cloud Security centre	System call graph	1-layer Autoencoder	Accuracy-93.68%
Xiao *et al*. [23]	Drebin project dataset	System calls	4-layer LSTM	Accuracy:93.7% Precision:0.913 Recall:0.966
Mishra *et al*. [19]	University of new maxico(UNM) dataset	System calls	CNN+LSTM	Accuracy-96.67%
Khater *et al*. [15]	ADFA-LD	System calls	1-laer MLP	Accuracy-94% Recall-0.95 F1-measure-0.92
HaddadPajouh *et al*. [11]	Arm based malware	Opcode sequence	3 different LSTM	Accuracy-98.18%
Darabian *et al*. [7]	Virus total threat intelligence platform	Opcode sequence	Adaboost	Accuracy-99.52% F1-measure-0.995
Azmood *et al*. [1]	ARM based Malware	Opcode sequence	Deep eigen space learning	Accuracy-99.68% Recall-0.983 Precision-0.985 F1-measure-0984
Proposed Model	IoTPOT malwares	System call and opcode sequence	Multichannel CNN(2, 1-layer CNN)	Accuracy-99.68% Recall-0.984 Precision-0.988 F1-measure-0.986

In Table 7, the detailed comparative summary between the existing and proposed solutions has been demonstrated. Among these existing solutions, only two of them [13, 23] are based on real IoT malware samples. The maximum accuracy achieved by the existing solution using system calls is 96.67%, whereas the maximum accuracy using opcode sequences is 99.6%. The proposed work using both system calls as well as opcode sequences is 99.68%. From these observations, it can be concluded that the proposed methodology outperforms the existing solutions.

7. Conclusions

Generally, IoT/smart devices face severe threats because of so many loopholes exist in their design. Since the IoT environment consists of elements that belong to diverse communication protocols, traditional malware detection techniques fail to detect attacks launched over the IoT in a reliable way. The key challenges of providing a solution to detect malicious samples in IoT devices are limited hardware resources, software capabilities, and speed. Hence, in this work, fog-based malware detection has been implemented by placing the malware engine in the fog nodes. The fog node is not only a replacement for the cloud but also very nearer to the edge devices in order to provide various services with minimum latency and bandwidth. In this work, the comparative analysis between deep learning and machine learning techniques has been carried out using various performance metrics. The deep learning architecture is designed with a single hidden layer and minimum number of epochs. Based on the analysis, the performance of the classification using multichannel CNN is higher than the Multichannel RNN, CNN and RNN is measured in terms of accuracy, precision, recall and F1-measure. In this work, a lightweight malware detection is built with the fog nodes to reduce the complexity of the IoT devices with an accuracy of 99.8% and recall of 0.998. This work can be further extended by adding more channels with respect to new features utilised additionally such that each channel works on each type of feature vector to detect new variants of malware samples. This solution relies on binary classification and it can be enhanced by implementing the same solution for multiclass classification.

References

[1] Azmoodeh A., Dehghantanha A., and Choo K., "Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning" *IEEE Transactions on Sustainable Computing*, vol. 4, no.1, pp. 88-95, 2018.

[2] Abbas M. and Srikanthan T., "Low-Complexity Signature-Based Malware Detection for Iot Devices," *in Proceedings of International Conference on Applications and Techniques in Information Security*, Nanning, pp. 181-189, 2018.

[3] Antonakakis M., April T., Bailey M., Bernhard M., Bursztein E., Cochran J., Durumeric Z., Halderman J., Invernizzi L., Kallitsis M., Kumar D., Lever C., Ma Z., Mason J., Menscher D., Seaman C., Sullivan N., Thomas K., and Zhou Y., "Understanding the Mirai Botnet," *in Proceedings of the 26th USENIX Security Symposium*, pp. 1093-1110, 2017.

[4] An N., Duff A., Naik G., Faloutsos M., Weber S., and Mancoridis S., "Behavioral Anomaly Detection of Malware on Home Routers", *in Proceedings of 12th International Conference on*

Malicious and Unwanted Software (MALWARE), Fajardo, pp. 47-54, 2017.

[5] An N., Duff A., Noorani M., Weber S., and Mancoridis S., "Malware Anomaly Detection on Virtual Assistants," *in Proceedings of 13th International Conference on Malicious and Unwanted Software (MALWARE)*, Nantucket, pp. 124-131, 2018.

[6] Breitenbacher D., Homoliak I., Aung Y., Tippenhauer N., and Elovici Y., "HADES-Iot: A Practical Host-Based Anomaly Detection System for Iot Devices," *in Proceedings of the ACM Asia Conference on Computer and Communications Security*, New York, pp. 479-484, 2019.

[7] Darabian H., Dehghantanha A., Hashemi S., Homayoun S., and Choo K., "An Opcod-Based Technique for Polymorphic Internet of Things Malware Detection" *Concurrency and Computation: Practice and Experience*, vol. 32, no. 6, pp. e5173, 2020.

[8] Devarajan R. and Rao P., "An Efficient Intrusion Detection System By Using Behaviour Profiling and Statistical Approach Model," *The International Arab Journal of Information Technology*, vol. 18, no. 1, pp. 114-124, 2021.

[9] Fleshman W., Raff E., Zak R.., McLean M., and Nicholas C., "Static Malware Detection and Subterfuge: Quantifying the Robustness of Machine Learning and Current Anti-Virus," *in Proceedings of 13th International Conference on Malicious and Unwanted Software (MALWARE)*, Nantucket, pp. 1-10, 2018.

[10] Gerber A. and Romeo J., "Connecting all the Things in The Internet of Things," IBM Corporation, pp. 1-10, 2017.

[11] HaddadPajouh H., Dehghantanha A., Khayami R., and Choo K., "A Deep Recurrent Neural Network Based Approach for Internet of Things Malware Threat Hunting," *Future Generation Computer Systems*, vol. 85, pp. 88-96, 2018.

[12] Hou S., Saas A., Chen L., and Ye Y., "Deep4maldroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graph," *in Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*, Omaha, pp. 104-111, 2016.

[13] Iwendi C., Jalil Z., Javed A., Reddy T., Kaluri R., Srivastava G., and Jo O., "Keysplitwatermark: Zero Watermarking Algorithm for Software Protection Against Cyber-Attacks," *IEEE Access*, vol. 8, pp. 72650-72660, 2020.

[14] Jeon J., Park J., and Jeong Y., "Dynamic Analysis for Iot Malware Detection with Convolution Neural Network Model," *IEEE Access*, vol. 8, pp. 96899-96911, 2020.

[15] Khater B., Wahab A., Idris M., Hussain M., and Ibrahim A., "A Lightweight Perceptron-Based Intrusion Detection System for Fog Computing," *Applied Sciences*, vol. 9, no.1, pp. 178, 2019.

[16] Kolosnjaji B., Zarras A., Webster G., and Eckert C.," Deep Learning for Classification of Malware System Call Sequences" *in Proceedings of Australasian Joint Conference on Artificial Intelligence*, Hobart, pp. 137-149, 2016.

[17] Khan M. and Salah K., "Iot Security: Review, Blockchain Solutions, and Open Challenges," *Future Generation Computer Systems*, vol. 82, pp. 395-411, 2018.

[18] Kim T., Kang B., Rho M., Sezer S., and Im E., "A Multimodal Deep Learning Method for Android Malware Detection using Various Features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773-788, 2018.

[19] Mishra P., Khurana K., Gupta S., and Sharma M., "VMAnalyzer: Malware Semantic Analysis using Integrated CNN and Bi-Directional LSTM for Detecting VM-level Attacks in Cloud" *in Proceedings of 12th International Conference on Contemporary Computing*, Noida, pp. 1-6, 2019.

[20] Pa Y., Suzuki S., Yoshioka K., Matsumoto T., Kasama T., and Rossow C., "Iotpot: A Novel Honeypot for Revealing Current Iot Threat," *Journal of Information Processing*, vol. 24, pp. 522-533, 2016.

[21] Shobana M. and Poonkuzhali S., "A Novel Approach to Detect Iot Malware By System Calls Using Deep Learning Techniques," *in Proceedings of International Conference on Innovative Trends in Information Technology*, Kottayam, pp. 1-5, 2020.

[22] Vinayakumar R., Alazab M., Soman K., Poornachandran P., Al-Nemrat A and Venkatraman S., "Deep Learning Approach for Intelligent Intrusion Detection System" *IEEE Access*, vol. 7, pp. 41525-41550, 2019.

[23] Xiao X., Zhang S., Mercaldo F., Hu G., and Sangaiah A., "Android Malware Detection Based on System Call Sequences and LSTM" *Multimedia Tools and Applications*, vol. 78, pp. 3979-3999, 2019.

[24] Yang Y., Wu L., Yin G., Li L., and Zhao H., "A Survey on Security and Privacy Issues in the Internet-of-Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250-1258, 2017.

[25] Zielonka A., Woźniak M., Garg S., Kaddoum G., Piran M., and Muhammad G., "Smart Homes: How Much Will They Support Us? A Research On Recent Trends And Advances," *IEEE Access*, vol. 9, pp. 26388-26419, 2021.

Shobana Manoharan is a research scholar at the department of computer science and Engineering at Rajalakshmi Engineering college. Her research interest includes network security, IoT, outlier mining, Artificial intelligence, NLP. She has published 10 technical papers in international journals and conference. Life time membership in IAENG, SDIWC, SEEE.

Poonkuzhali Sugumaran Professor in the Department of Computer Science and Engineering and Head of Centre for Assistive Devices and Technologies has been with Rajalakshmi Engineering College since August 2000. Her area of specialization is Web Mining, Outlier mining, Information Retrieval, Knowledge Management, Big Data Analytics and E-Learning. Authored 6 texts books and published more than 75 papers in various reputed conferences and in international journals. Received 6 Best Paper Awards for the oral paper presentation. Received International Paper Presenter Award from Computer Society of India She is doing consultancy work for Sentinel Radiologist Solutions and Skill Council for Person with Disability.

Kishore Kumar Artificial Intelligence (AI) Engineer, VisioNxt a Mega Project under R&D Division of Ministry of Textiles. He completed Doctoral Studies (PhD) in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, West Bengal. His area of specialization is Computer Vision, Speech Processing, Web Service Security, Web Mining, Information Retrieval. Published more than 10 papers in various reputed conferences and in international journals. Received one Best Paper Awards for the oral paper presentation. Received Grants from Tamil Nadu State Council for Science and Technology for presenting papers in the International conference for oral presentation. He is a reviewer for the Neural Processing Letters journal.