Malware Detection in Industrial Internet of Things based on Hybrid Image Visualization and Deep Learning Model

Article *in* Ad Hoc Networks · May 2020

DOI: 10.1016/j.adhoc.2020.102154

CITATIONS
7

READS
258

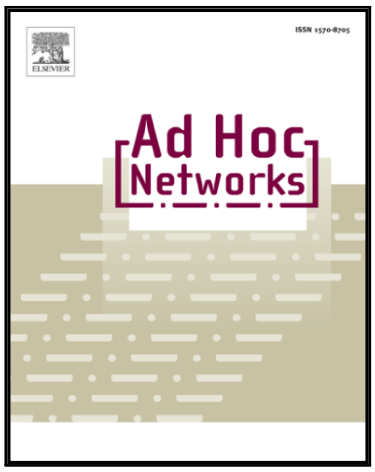
7 authors, including:

Hamad Naeem
Neijiang Normal University
27 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Farhan Ullah
Sichuan University
33 PUBLICATIONS **255** CITATIONS
SEE PROFILE

Muhammad Rashid Naeem
Sichuan University
22 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Shehzad Khalid
Bahria University
93 PUBLICATIONS **1,627** CITATIONS
SEE PROFILE



Malware Detection in Industrial Internet of Things based on Hybrid Image Visualization and Deep Learning Model

Hamad Naeem , Farhan Ullah , Muhammad Rashid Naeem , Shehzad Khalid , Danish Vasan , Sohail Jabbar , Saqib Saeed

PII: S1570-8705(19)30694-8
DOI: https://doi.org/10.1016/j.adhoc.2020.102154
Reference: ADHOC 102154

To appear in: *Ad Hoc Networks*

Received date: 20 July 2019
Revised date: 19 February 2020
Accepted date: 24 March 2020

Please cite this article as: Hamad Naeem , Farhan Ullah , Muhammad Rashid Naeem , Shehzad Khalid , Danish Vasan , Sohail Jabbar , Saqib Saeed , Malware Detection in Industrial Internet of Things based on Hybrid Image Visualization and Deep Learning Model, *Ad Hoc Networks* (2020), doi: https://doi.org/10.1016/j.adhoc.2020.102154

Malware Detection in Industrial Internet of Things based on Hybrid Image Visualization and Deep Learning Model

Hamad Naeem[1], Farhan Ullah[2,3], Muhammad Rashid Naeem[2], Shehzad Khalid[4], Danish Vasan[5], Sohail Jabbar[6*], Saqib Saeed[7]

[1] School of Computer Science, Neijiang Normal University, Neijiang, Sichuan, P.R China, 641100
[2] College of Computer Science, Sichuan University, Chengdu, 610065, China
[3] Department of Computer Science, COMSATS University Islamabad, Sahiwal Campus, Sahiwal 57000, Pakistan
[4] Department of Computer Engineering, Bahria University, Islambad Pakistan
[5] School of Software Engineering, Tsinghua University, Beijing, PR. China
[6] CfACS IoT Lab, Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, United Kingdom
[7] Department of Computer Information Systems, College of Computer Science and Information Technology, Imam Abdurrahman Bin Faisal University, P.O. Box 1982, Dammam, Saudi Arabia

[1]hamadnaeemh@yahoo.com, [2]farhankhan.cs@yahoo.com, [1]rashidnaeem717@yahoo.com , [3]shehzad_khalid@hotmail.com, [4]danish.wasan@gmail.com [5]sjabbar.research@gmail.com, [6]sbsaed@iau.edu.sa

Corresponding author: Sohail Jabbar (sjabbar.research@gmail.com)

Abstract

Now the Industrial Internet of Things (IIoT) devices can be deployed to monitor the flow of data, the source of collection and supervision on a large scale of complex networks. It implements large networks for sending and receiving data connected by smart devices. Malware threats, which are primarily targeted at conventional computers linked to the Internet, can also be targeted at IoT machines. Therefore, a smart protection approach is needed to protect millions of IIoT users against malicious attacks. On the other hand, existing state-of - the-art malware identification methods are not better in terms of computational complexity. In this paper, we design architecture to detect malware attacks on the Industrial Internet of Things (MD-IIOT). For an in-depth analysis of malware, a methodology is proposed based on color image visualization and deep convolution neural network. The findings of the proposed method are compared to former approaches to malware detection. The experimental results indicate that the proposed method's predictive time and detection accuracy are higher than that of previous machine learning and deep learning methods.

Keywords: Image Visualization; Deep Learning; Industrial Internet of Things; Malware Analysis

1. Introduction

Currently, the Internet is a major source of massive information and interaction. Because of online services, e-shopping, social networking and e-banking, the threats of malicious attacks are more challenging. The hackers are attempting to harm the privacy of smartphones, personal computers and IoT systems [1]. The cloud service companies provide a network-operating connection between computers, smart phones and IoT devices. Today, thanks to intelligent cloud services, end-users can connect more quickly and easily. Such services may be designed to protect the linking devices against malicious attacks.

There are different types of malware detection techniques for window based malware analysis, however mostly use certain signatures to smell the malware attacks. Mainly, malware detection techniques are divided into two types, i.e. static and dynamic analysis. The real-time behavior of malware may be sense using virtual environment in dynamic analysis. It is further divided in to different types of techniques, i.e. function call observing, function parameter analysis, information flow tracking, instruction traces and dynamic visual analysis. Various online services available to accomplish dynamic investigation of malware such as TT analyzer, Anubis and CW Sandbox. Dynamic features extraction of malware is very hard and time consuming to catch the binary information of malware as compared to static analysis [2-3].

Static analysis doesn't need code execution in a virtual environment and it is an efficient way to mine the structured information of malware executable. The signature based malware detection perform static analysis and it is further divided into different types of technique, i.e. string signature, Opcode frequency distribution, control flow graph and n-gram. These methods use the disassemblers, i.e. IDA Pro [4] and OllyDbg [5], to uncover the malware executable and then apply static analysis for feature extraction. This secret information is very supportive to get the hidden strings from malware executable. Such as, n-byte sequence features can be captured from byte sequence method. The function call graph is a type of static analysis which is used to retrieve the structural features from malware executable. The control structure patterns can also be retrieved from dissembled data using Opcode sequence [6]. Further, the binary conversion method can be used to extract the distinctive useful binary information from source code structures. Moser et al. [7] clearly stated that static analysis is not commendable of binary conversion analysis as these methods first, uncover malware and then apply static analysis.

Beside of this, the important drawback of these techniques is the immense processing costs.

The sole purpose of the using static analysis is to decrease the resource and time costs. Further, the static visual analysis is used extensively to transform the format information such as PE headers, executable code and meta data to image. Subsequently, the graph entropy [8], image matrices [9] and image processing techniques are used capture the hidden similar patterns for malware detection [10-11]. The algorithms based on machine learning are very dominant to capture malware patterns form executable files. It uses intelligent representation methods such GIST, PCA (Principal Component Analysis), DWT (Discrete Wavelet Transform), SURF (Speeded Up Robust Feature) and SIFT (Scale Invariant Feature Transform) to extract the image texture features from malware. The DWT, GIST and PCA techniques may be used to capture global background structural patterns from malware images. Opposing this, the SURF and SIFT techniques capture local information from image contents. The static visual analysis is fast and less resource consuming as it does not care of reverse engineering part. But still it is unproductive to capture critical information of malware image. Presently, malware authors are continuously creating new signatures IIOT and making their detection more challenging. Beside this, the current solutions require large resources and high computational time for malware detection in IIOT. This paper brings the following contributions:

1. Architecture is proposed for the detection of malware attacks in the Industrial Internet of Things (MD-IIOT).
2. A hybrid approach focused on color image visualization and deep convolution neural network is developed for in-depth malware analysis.
3. An optimized deep convolution neural network is proposed which is cost-effective and scalable computationally, with low run-time costs.
4. A comparison is made between the proposed method and the previous methods to malware detection. According to experimental results, the proposed method is stable, more reliable and less resource intensive.

The rest sections of this paper are arranged as follows. Section 2 explains existing works, section 3 describes the architecture model for malware detection in IIOT, and section 4 provides the experiment results and discussion. Finally, conclusions are discussed in Section 5.

2. Existing Works

More studies are conducted for malware's feature visualization to get the high accuracy for

classification, performance, time and resource costs. To extract useful malware features, static analysis does not care of program execution. The structural information of malware such as executable code, meta data, and bitmaps may be transform in grayscale visual information in static analysis. Presently, there are different visualization methods i.e. image similarity matrix, graph similarity and other image processing technique are designed to identify visual malware variants.

Eul et al. [8] recommended a graph-based relationship technique to classify malware features. First, these techniques are applied to covert hidden pattern of malware to gray scale images and after, different types of entropy graphs are generated using these images. The experimental results exposed that this technique attained 97.9% classification accuracy using dataset of 1000 samples from 50 distinctive malware families. Jae et al. [18] proposed image similarity method for detection of malware hidden patterns. They used Opcode sequence method to produce RGB colored pixels' information. Further, the results signified that the proposed method got 98% accuracy using 50 samples of 10 dissimilar families.

Barath et al. [13] designed a malware detection method which is used to extract texture patterns based on PCA method. Then, the PCA information are used as input to nearest neighbor algorithm to get better classification rate. The results indicated that the proposed approach got 96% accuracy using dataset of 10,000 samples taken from 8 malware families. Kesav et al. [14-15] shown that hig quality malware features can be extracted using machine learning classification. The proposed method is used to extract gray scale images from malware and then texture features are analyzed from these images. The authors applied Support Vector Machine (SVM) to get high similarity rate among these texture features. Ban et al. [11] designed an approach that is used to extract local malware image features using SURF technique. Further, the fingerprint matching technique is used to mine the local sensitive hashing patterns.

The proposed idea reached 85% prediction accuracy using dataset of 8410 samples collected from 25 malware groups. Nataraj et al. [10], the author designed an approach to classify malware based grayscale and GIST features. First, the malware is transformed into grayscale visualization and then GIST patterns are produced to classify each malware in corpus. The proposed idea got an accuracy of 97% using dataset of 9339 samples and these samples are collected from 25 malware families. Aziz et al. [19], the GIST technique is used to extract

texture features and after, feed forward neural network is used classify each malware from the corpus. This method achieved the classification results with 98% accuracy using dataset of 1710 samples collected from 8 malware families. Hamad et al. [31-32] first extracted hybrid local and global features of malware image and then applied the machine learning classification algorithm. For large scale analysis, their methods obtained 98.4% classification accuracy, whereas picking 9339 samples from 25 malware families of Malimg dataset. For small-scale analysis, their approaches received 99.21% classification accuracy, whereas selecting 5288 samples from 8 malware families of Malimg dataset. Mahmoud et al. [20], the author designed CNN model to classify malware from binary executable corpus. Further, this model gave 98.52% classification accuracy using dataset of 9339 samples from 25 malware executable. Moreover, this model is used to randomly select 10% samples in each cycle to test the family of a malware. Rajesh et al. [21], proposed malware classification model based on CNN. This model got an accuracy of 98% from dataset of 9339 samples. The random process is used to select 10% of samples in each cycle to test the family of the selected malware. Zhihua et al. [22] designed malware classification model using CNN. This approach gave an accuracy of 94.5% using corpus of 9339 samples taken from 25 different malware families. Farhan et al. [34] designed a deep convolutional neural network to detect malware attacks on the Internet through color image visualization. Their experimental results showed improved classification performance to measure cyber security threats. A scheme based on Random Coefficient Selection and Mean Modification Approach (RCSMMA) was proposed by Nasir et al. [35]. RCSMMA results in good performance for various common cyber-attacks. Fadi et al. [36] highlighted the main smart city applications and addressed the major issues of privacy and security in the application architecture of the smart cities due to malware attacks. D. Deebak et al. [37] proposed a secure routing and monitoring protocol with multi-variant tuples to prevent adversaries in the global sensor network. Fadi et al. [38] proposed a framework for addressing big data delivery issues in cloud-based IoT environments. Results revealed that the distributed algorithm outperformed the centralized one.

Our model is composed of four subsystems relative to other methods: (1) Malware Binary Visualization, (2) CNN fine tuning, (3) image normalization, and (4) choose Deep CNN architecture. Our approach covered both classification accuracy and overhead run-time in large data based on IIOT.

3. Malware Detection for Industrial Internet of Things (MD-IIOT)

In this paper, we design architecture of malware detection for Industrial Internet of Things (MD-IIOT), as shown in Fig.1.We install three databases in cloud storage, namely, traffic, behavior and log databases.

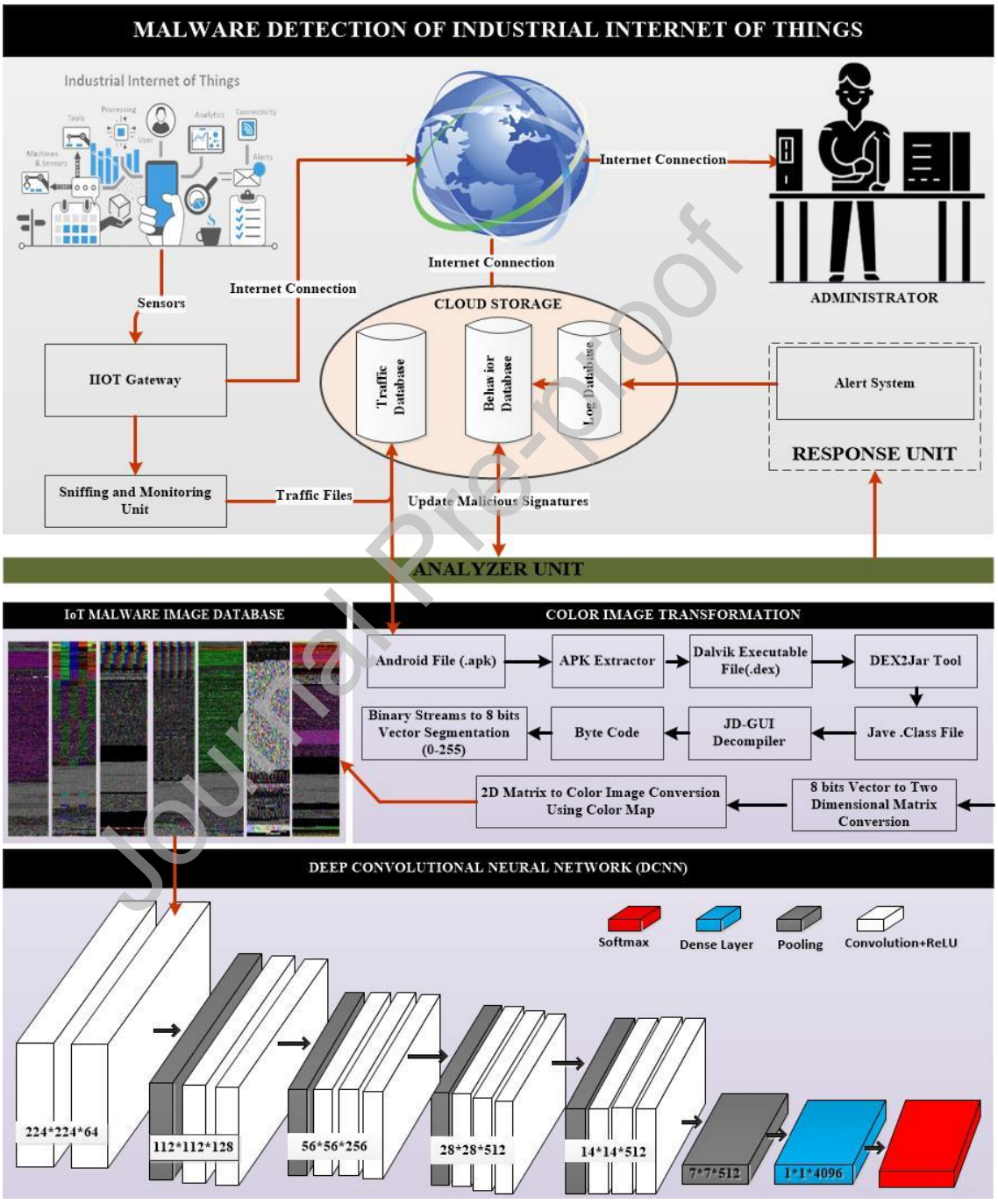


Figure 1. Architecture of Malware Detection for Industrial Internet of Things (MD-IIOT)

Traffic database collects the raw android files of IIOT using sniffing and monitoring unit. Behavior database consists of a list of datasets that represent the network history. Log database collects new malware fingerprints and updates behavior database regularly.

A large volume of data handling needs more time and large resources. Therefore in the analyzer unit, we design a Deep Convolutional Network (DCNN) along with color image transformation technique. The raw Android file is first converted into a color image and then passes through a DCNN model. The analyzer unit matches the file fingerprints with a behavior database and then classifies it as malware or benign. If any malicious activity is observed in-network, then response unit sends a final warning to the system administrator to take a suitable step. The detail workflow of each part of proposed architecture is described below.

3.1. Sniffing and Monitoring Unit

The sniffer is designed in the gateway to observe and gather information regarding incoming and outgoing traffic. It may be embedded in the software over a network to sense the sent and received data packets information. These packets are then stored in the raw traffic database. There are three databases, i.e. raw traffic, log and behavior, configured which are used to smell the ongoing activities over a network. These databases are installed in the cloud storage, i.e. fog computing storage. First, it stores the raw data gathered from the sniffer over a network and second part comprises the list of previously used datasets. These datasets are used to observe the profile history of the routers. The third part saves the signatures of the identified attacks. These different types of attacks are then used to constantly feed the behavior database.

3.2. Analyzer Unit

This is vital part of proposed architecture which consists of two modules, namely, color image transformation and DCNN model.

3.2.1.Color Image Transformation

For color image visualization, we unzip .apk file using extractor. File in .apk format typically contains a Class.DEX file that encapsulates all Dalvik bytes code. The structure of the DEX file consists of multiple fragments, namely, DEX header, String _ id, type _ ID, Proto _ ID field _ id, method_id, Class_Def, and data. The DEX header records the physical offset of the

other six data structures in the DEX file. We obtain byte code from .apk file in three steps. First, we decompress the apk file and received the class.Dex file. Second, we convert the class.Dex file into Java.Class file using the dex2jar tool. Third, we use JD-GUI decompile to extract byte code from Java.class file. Fig. 2 shows the corresponding relationship between the structure of a DEX file and the byte code image [1].

The nature of malware images is different from ordinary scene images. An ordinary scene image contains patterns in continuous form, but on another end malware image contains patterns in an arbitrary form. Therefore, deep learning techniques are quite promising to detect such types of patterns. Color images are more feature enrich than grayscale images which have only 256 colors. Also, large prominent features of malware binary can facilitate in classification performance of malware families. Deep learning has shown better results with large image datasets and features. Unlike machine learning, deep learning can apply filters to reduce noise automatically. The use of color images produces better results with deep learning algorithms. Therefore, we apply color image transformation for IIOT malware detection.

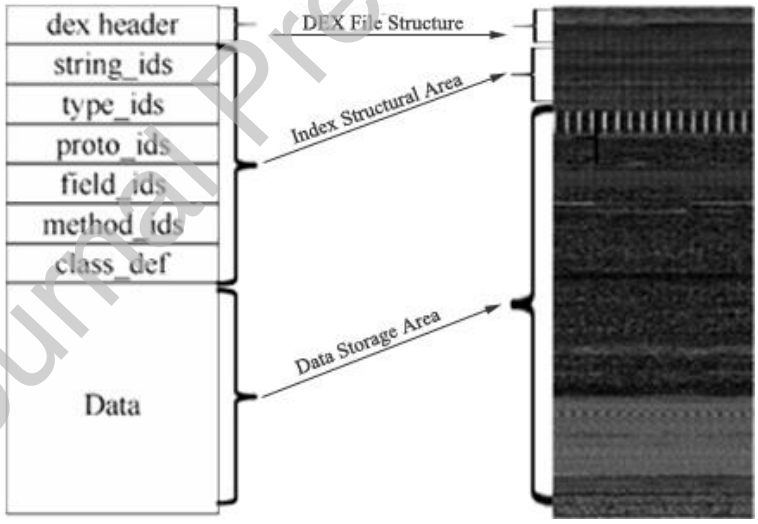


Figure 2. The correspondence between DEX file structure and byte code image [23]

The methodology of malware binary file to color image transformation consists of three significant steps. First, the malware binary bit string is divided into substrings. Each of substring is 8 bits in length and represents as a pixel. Each eight bit is considered as an unsigned integer (0-255). Second, the malware binary bit string is converted into a one-dimensional vector of decimal numbers. Third, the one-dimensional vector of decimal numbers is transformed into a

two-dimensional color matrix of the specified width. Based on empirical observations, Table.1 presents the image widths for different file sizes. The color image visualization for IIOT malware APK is presented in Fig.3.

Table 1. Image width for various file sizes [10]

File Size	Image width	File size	Image width
<10 KB	32	100 KB~200KB	384
10 KB~30 KB	64	200 KB~500KB	512
30 KB~60 KB	128	500 KB~1000KB	768
60 KB~100 KB	256	>1000 KB	1024

Figure 3. A Chunk of color images generated from IIOT malware APK file

3.2.2.Deep Convolutional Neural Network (DCNN)

In this study, we present in-depth analysis using DCNN model. The structure of proposed DCNN model consists of four components, as shown in Fig.1. Input layer brings training images into the model. Convolutional layer first reduces the noise and enhances signal characteristics. The performance of proposed deep learning model is enhanced with optimization of Convolutional kernel width, the number of hidden units and learning rate. Secondly, pooling layer reduces the amount of data by holding meaningful information. Thirdly, dense layer converts the two-dimensional features into one-dimensional features and further supplies them for classification. Additional, fine tuning number of neurons with activation function and learning error rate in different layers also increase the classification performance. Finally, the classifier identifies the images as malware or benign. The brief description of each layer is presented below.

(1) Convolution Layer

Convolution layer decrease total number of parameters and captures important features, namely, interpretation, rotation, and scale invariance. It significantly reduces over fitting

problem and improves the generalization capacity of DCNN model. The input of convolutional layer consists of several maps [24]. The total addition of convolution values of input maps shows the value of output map. The following equation presents it:

$$x_j^l = f\left(\sum_{i \in M_j} x_j^{l-1} * k_{ij}^l + b_j^l\right) \tag{1}$$

Where M_j denotes the group of input maps; k_{ij}^l represents convolution kernel that is used for combine the i^{th} input with j^{th} output feature map; b_j^l shows the bias corresponding to the i^{th} feature map, and f is the activation function.

$$\delta_j^l = \delta_j^{l+1} W_j^{l+1} o f'\left(u^l\right) = \beta_j^{l+1} up\left(\delta_j^{l+1}\right) o f'\left(u^l\right) \tag{2}$$

Where *l+1* layer shows the pooling layer; *W* denotes the convolution kernel, and β_j^{l+1}o *up*(.) does up sampling. The partial derivative of the error cost function and convolution kernel is presented as follows:

$$\frac{\partial E}{\partial b_j} = \sum_{s,t}\left(\delta_j^l\right)s,t \tag{3}$$

$$\frac{\partial E}{\partial k_{ij}^l} = \sum_{s,t}\left(\delta_j^l\right)s,t\left(p_i^{l-1}\right)s,t \tag{4}$$

Where $\left(p_j^{l-1}\right)u,v$ shows the patch for each convolution of x_i^{l-1}; $k_{ij,}^l(u,v)$ denotes the centre of the patch.

(2) Pooling Layer

Pooling layer applies two types of pooling, namely, maximum and average pooling. It does not affect from backward propagation. It reduces the effect of image deformation in DCNN model. It enhances the performance of model and reduces total number of feature map's dimensions. The output of each sampling is provided with feature map:

$$x_j^l = f\left(down\left(x_j^{l-1}\right) + b_j^l\right) \tag{5}$$

Where *down (.)* does a pooling task, and *b* shows bias value. The value of sensitivity is calculated by using the following equation:

$$\delta_j^l = \delta_j^{l+1} W_j^{l+1} o f^{'}\left(u^l\right) \tag{6}$$

The partial derivative of the error cost function with respect to bias *b* is presented as follows:

$$\frac{\partial E}{\partial b_j} = \sum_{s,t}\left(\delta_j^l\right) u, v \tag{7}$$

(3) Dense Layer

The output of pooling layer is further classified by using a dense layer. The neuron of a dense layer connects with each neuron in the pooling layer. It first transforms a two-dimensional feature vector into a one-dimensional feature vector, and then further supplies it to an output layer.

(4) Output Layer

Android samples are identified as malware or benign. For data training, the DCNN model applies Softmax-Cross-Entropy loss. The training data loss *k* is defined as follows:

$$Loss = -\log\left(\frac{\exp(f_{zt})}{\sum_k \exp(f_{zt})}\right) \tag{8}$$

Where f_{zt} represents rank of k^{th} class; and f_{zt} represents score of correct family. Adam optimizer is used to learn parameters of the model, which reduces the training data loss.

3.3. Response Unit

It comprises the alert method and log database. The analyzer unit gives signal to the system's admin for taking the specific action, if any irregular movement occurs in the network. It stores the new signature of the malicious activity in the log database and then the specified signature may be used to feed the profile behavior database.

4. Experimental Results and Discussion

4.1. Experimental Dataset, Design and Evaluation

A publically available IIOT dataset, namely, Leopard Mobile dataset was selected for performance evaluation of proposed malware detection method. The dataset contained 14,733 malware samples and 2486 benign samples of different IIoT applications, namely, weather,

business, finance and travel. The dataset was collected from IKM Laboratory[1].

Beside this, a publicly available Windows dataset, namely, the Malimg dataset was used for comparison of classification performance among former methods and proposed malware detection method. Malimg dataset contained 9339 samples of 25 malware families. The dataset obtained from vision research lab of University California[2].

The experiments performed on GPU version GTX1080 NVIDIA, the RAM was 758GB, the operating system was 64-bit Ubuntu 16.04, and Python Tensor flow 1.9 developed the DCNN model. TensorFlow provides a flexible environment to design deep learning models using hyper parameters. It performs operations by using multi-dimensional arrays. It does parallel execution to speed up the classification process [33]. The parameters of VGG-16 [25] were not initialized in DCNN model.

First, we randomly selected 70% of the training data and 30% of the test data for experimentation. Previously, several researchers [26-27] recommended that 70%~80% training data ratio is more suitable choice for experimentation. Second, we performed experiments with two different malware image sizes, i.e. 224x224 and 229x229. Third, we selected three evaluation metrics, namely, precision, recall, and accuracy for performance evaluation. The number of True Positives (TPs) and False Positives (FPs) represented the number of malware samples classified as false and true, respectively. Similarly, the number of True Negatives (TNs) and False Negatives (FNs) represented the number of benign samples classified as false and true, respectively.

$$Precision = \frac{TP}{TP + FP} \quad , \quad Recall = \frac{TP}{TP + FN} \tag{9}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$F\text{-}measure = \frac{2 * TP}{2TP + FP + FN} \tag{11}$$

[1] https://sites.google.com/site/nckuikm/home
[2] https://vision.ece.ucsb.edu/research/signal-processing-malware-analysis

4.2. Performance Analysis for Proposed IIOT Malware Detection

We measured the impact of different image ratios, i.e. 224×224 and 229×229 on the classification performance of the proposed method. We selected Leopard Mobile dataset for performance evaluation in this section. We obtained maximum classification performance of our method on 224×224 image ratio. We also evaluated the performance of proposed method on 229×229 image size. However, the difference between 224×224 and 229×229 image sizes was response time. Hence, we decided that 224×224 image ratio is more suitable option for our method. The dynamic graph for accuracy, validated accuracy, loss and validated loss for both image ratios is shown in Figs. 4 and 5.

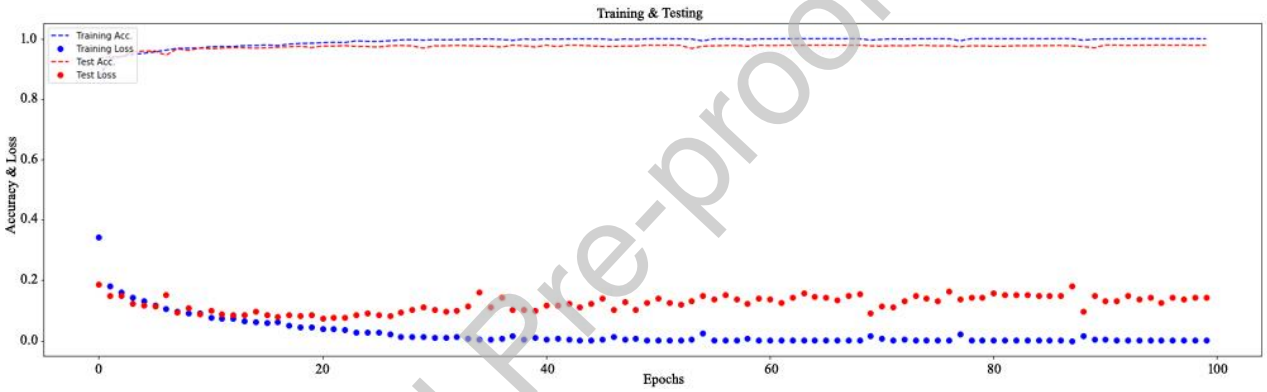


Figure 4. Leopard Mobile dataset-Dynamic graph for accuracy validated accuracy, loss and validated loss of proposed method (Image Ratio: 224×224)

Figure 5. Leopard Mobile dataset-Dynamic graph for accuracy validated accuracy, loss and validated loss of proposed method (Image Ratio: 229×229)

Tables 2 demonstrate the impact of different image ratios on classification performance through overall and class wise comparison. 224×224 image ratio obtained 97.81% overall testing

accuracy with 20 Sec computational time. 229×229 image ratio received 97.46% overall testing accuracy with 36 Sec computational time. Fig. 6 (a-b) show the confusion matrices for 224×224 and 229×229 image ratios. Here, the ground truth means true classes and predicted classes of a dataset after applying the proposed DCNN. Compared to Fig.6 (b), Fig.6 (a) displayed that our method less suffered from misclassification on 224×224 image ratio.

Figure 6(a). Leopard Mobile dataset- Confusion matrix for image ratio: 224×224

Figure 7(b). Leopard Mobile dataset- Confusion matrix for image ratio: 229×229

Table 2. Performance Analysis of proposed malware detection for different image ratios (IIOT Malware)

Leopard Mobile Malware Dataset						
Over-All Performance						
Image Ratio	**Precision (%)**		**Recall (%)**	**F-measure (%)**	**Accuracy (%)**	**Time (sec)**
224×224	95.16		95.10	95.13	97.81	20s
229×229	95.39		94.22	94.80	97.46	36s
Class-Wise Performance						
	Malware			**Benign**		
Image Ratio	**Precision (%)**	**Recall (%)**	**F-measure (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
224×224	97.36	96.90	97.13	82.13	84.45	83.27
229×229	98.26	98.77	98.52	92.53	89.67	91.08

4.3. Comparison Of Classification Performance Among Former Methods And Proposed Method

In this section, we first measured the effect of varying malware image dimensions i.e. 224×224 and 229×229 on the output of proposed solution and then compared its classification performance with former methods. We selected 9339 malware samples from 25 families of Malimg dataset to achieve this goal. The dynamic graph for accuracy, validated accuracy, loss and validated loss for both image ratios is shown in Figs. 7 and 8. The difference between both image ratios was prediction time. For 224×224 image ratio, our method achieved 98.47% overall testing accuracy with 17 Sec computational time. On other end, our method obtained 98.79% overall testing accuracy with 30 Sec computational time for 229×229 image ratio. Fig. 9(a-b) show the confusion matrices for 224×224 and 229×229 image ratios. Compared to Fig.9 (a), Fig.9 (b) display that our method less suffered from misclassification on 229×229 image ratio.

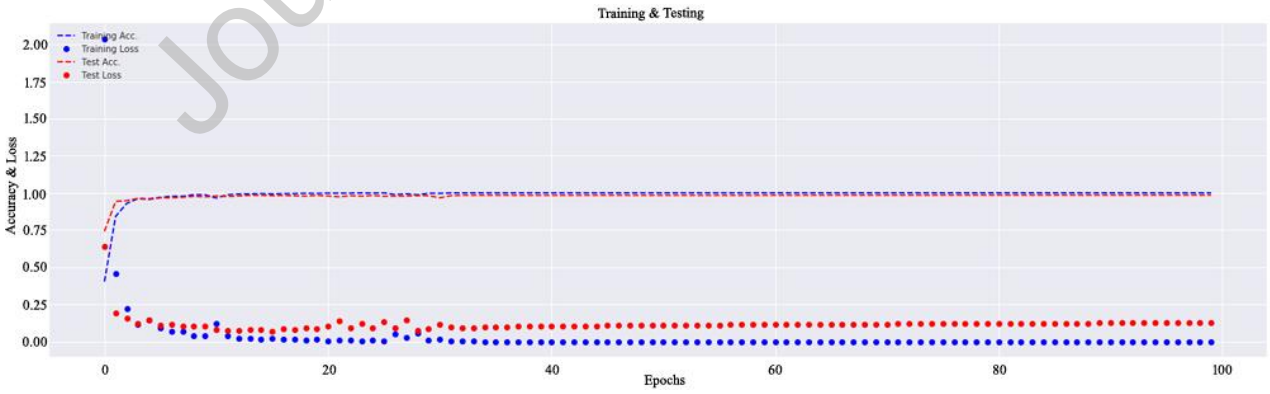


Figure 8. Malimg dataset-Dynamic graph for accuracy validated accuracy, loss and validated loss of proposed method (Image Ratio: 224×224)

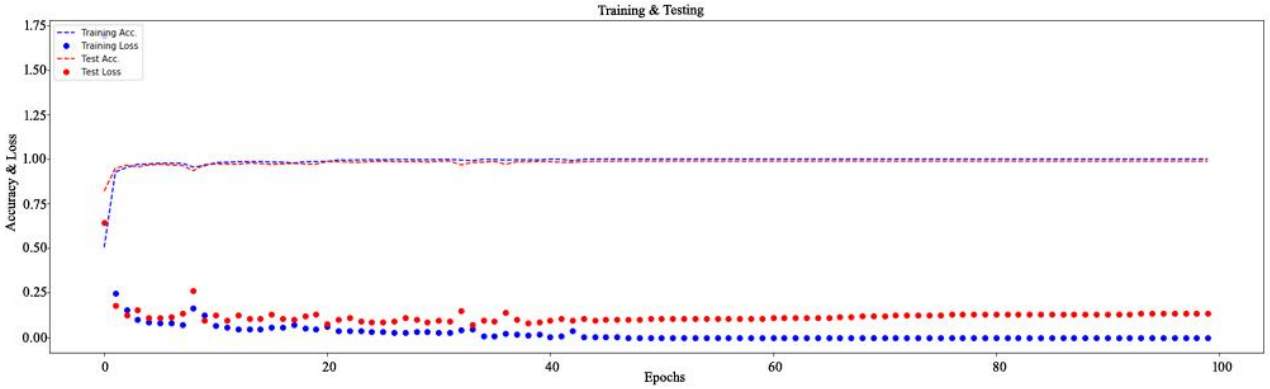


Figure 9. Malimg dataset-Dynamic graph for accuracy validated accuracy, loss and validated loss of proposed method (Image Ratio: 229×229)

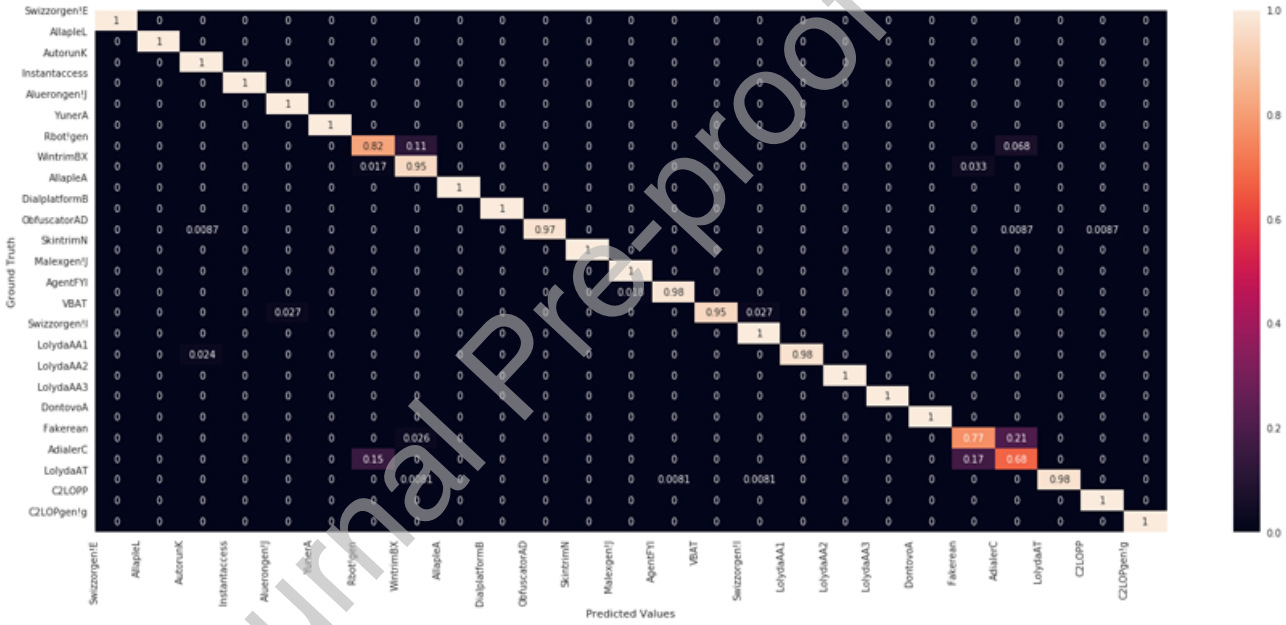


Figure 10(a). Malimg dataset-Confusion matrix for image ratio: 224×224

Table 3 exhibits the performance of our method for individual malware family. The experimental results indicated that proposed DCNN method achieved maximum classification performance for most of malware families on both images ratios (224×224 and 229×229). However in our case, 229×229 image ratio is quite suitable for malware detection with less false positive rate.

To compare our model, we prepared our baseline for comparison with previous state-of-the-art malware detection techniques. First experiment, we extracted texture features from malware color images using GIST descriptor [10].

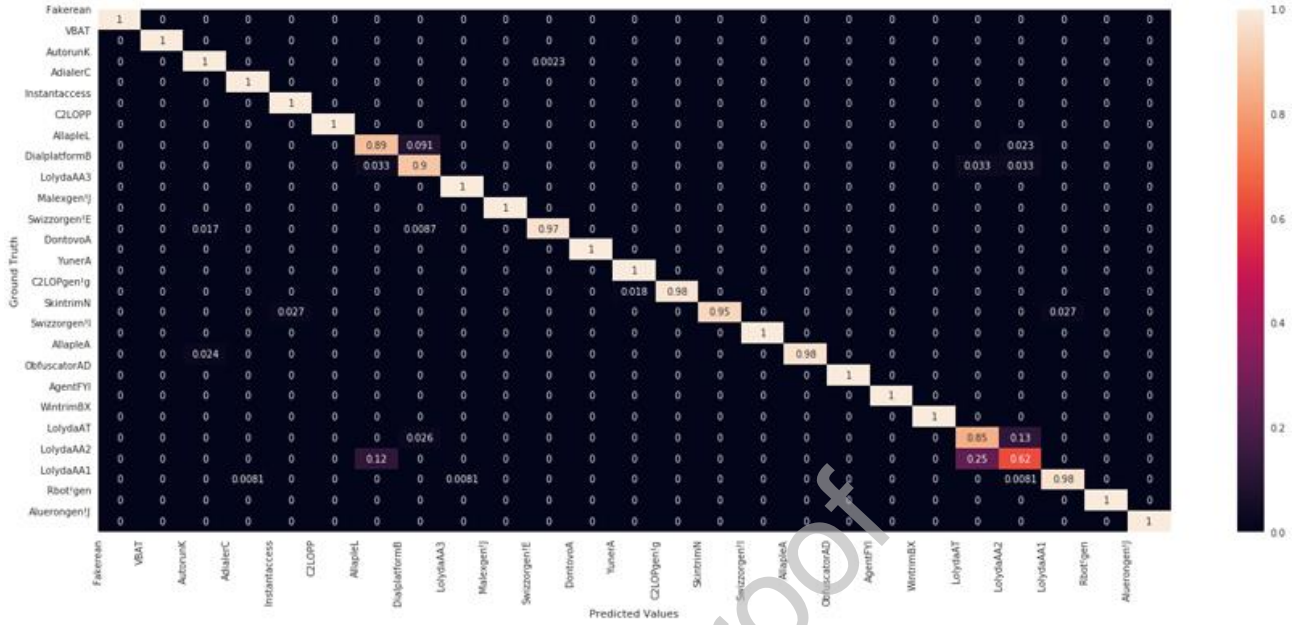


Figure 11(b). Malimg dataset-Confusion matrix for image ratio: 229×229

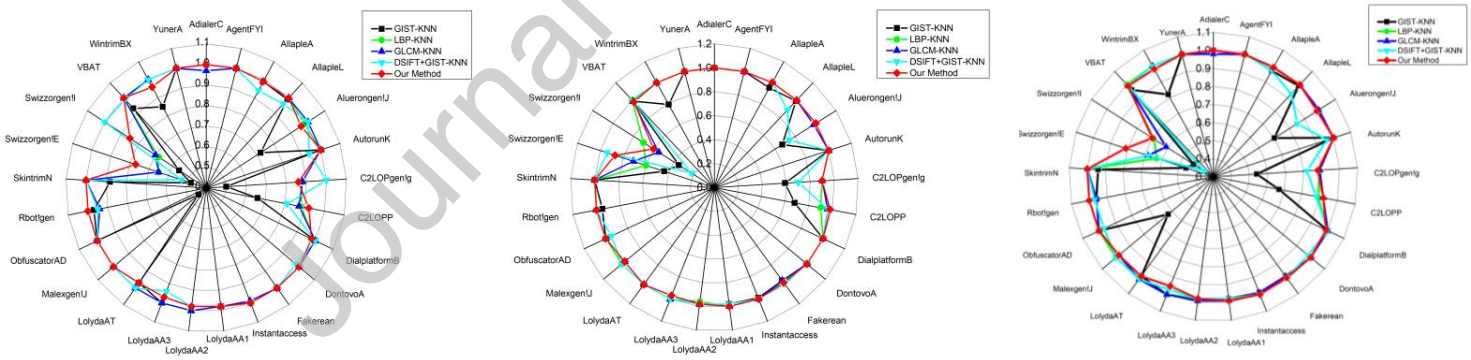
Table 3. Performance Analysis of proposed malware detection for different image ratios (Windows Malware)

Malimg Windows Malware Dataset						
Malware Families	**224×224**			**229×229**		
	PR (%)	**RR (%)**	**FM (%)**	**PR (%)**	**RR (%)**	**FM (%)**
AdialerC	100	100	100	100	100	100
AgentFYI	100	100	100	100	100	100
AllapleA	99.77	100	99.88	99.88	100	99.94
AllapleL	100	100	100	100	100	100
Aluerongen!J	98.36	100	99.17	96.77	100	98.36
AutorunK	100	100	100	100	100	100
C2LOPgen!g	83.72	81.81	82.75	86.95	90.90	88.88
C2LOPP	89.06	95	91.93	92.18	98.33	95.16
DialplatformB	100	100	100	98.18	100	99.08
DontovoA	100	100	100	100	100	100
Fakerean	100	97.39	98.67	100	98.26	99.12
Instantaccess	100	100	100	100	100	100
LolydaAA1	98.46	100	99.22	98.46	100	99.22
LolydaAA2	98.21	98.21	98.21	98.21	98.21	98.21
LolydaAA3	100	94.59	97.22	97.22	94.59	95.89
LolydaAT	96	100	97.95	97.95	100	98.96
Malexgen!J	100	97.56	98.76	100	97.56	98.76
ObfuscatorAD	100	100	100	100	100	100
Rbot!gen	100	100	100	100	100	100
SkintrimN	100	100	100	100	100	100
Swizzorgen!E	76.92	76.92	76.92	77.27	87.17	81.92
Swizzorgen!I	69.23	67.5	68.35	85.71	60	70.58

VBAT	100	97.56	98.76	100	98.37	99.18
WintrimBX	96.77	100	98.36	96.77	100	98.36
YunerA	100	100	100	100	100	100
Average	**98.47**	**98.47**	**98.46**	**98.79**	**98.79**	**98.75**

Note: *FM=F-Measure, PR=Precision Rate, RR=Recall Rate*

GIST [10] is a valid feature descriptor that has been widely used in the existing study. Second experiment, we extracted local binary patterns from malware color images using LBP descriptor [28]. Third experiment, we extracted texture features from malware color images by using GLCM descriptor [22]. GLCM provided perfect texture features of malware images. Fourth experiment, we extracted hybrid local and global features from malware color images. We combined Dense SIFT [17] and GIST [12] feature descriptors for malware identification. We selected 512-dimensional GIST, LBP, D-SIFT+GIST, and 20-dimensional GLCM features for experimental purpose, The K-Nearest Neighbor (K-NN) used to classify the malware images. The average classification rate of our method was 98.79%, which was highest among all malware classification techniques. We also compared our proposed model with conventional malware detection researches. The author [22] stated maximum performance with their machine learning and deep learning models. Similarly, the author [29] mined textural features of malware image using DCNN model.



(a) Precision **(b) Recall** **(c) F-measure**

Figure 12(a-c). Precision, Recall and F-measure of different malware classification algorithms in 25 malware families.

The author [30] extracted malware gray scale image features using GRU-SVM model. Fig.10 (a-c) shows that our method found superior in all of the statistical measures for all malware families, even in family Swizzorgen!E and Swizzorgen!I where other algorithms struggle. For evaluating obfuscation resistance, our analysis is focused only on the polymorphic obfuscation/packing, and

selecting 142 samples in the ObfuscatorAD family of Malimg dataset. The experimental result validated that our proposed method is resistance to polymorphic obfuscation/packing, as well as showed that texture-based malware detection is effective. Since there are only 142 samples in the family ObfuscatorAD, most of the algorithms cannot identify ObfuscatorAD very well. It is worth noting that our proposed method keeps excellent performance (100%). Table 4 shows the Accuracy, Precision, Recall rate and time of the different algorithms. In terms of classification accuracy, Precision, and Recall rate, our method achieved 98.79%, which was better than the other machine learning and deep learning malware classifications. The result suggests the capability of our method to find correspondences between similar visual contents and allows the malware analysis to classify malware and identify variants. Although the classification time of our method was from 17s to 30s, which is slightly more than other algorithms, still it was more accurate to process small-scale and large-scale malware analysis.

Table 4.The comparison of classification performance among former methods and proposed method

Methods	Year	Image Size	Technique	PR (%)	RR (%)	CA (%)	Time
GIST-KNN	-	192×192	Machine Learning	**89.35**	**87.98**	**89.35**	**10.2s**
LBP-KNN				**96.20**	**96.31**	**96.20**	**8.36s**
GLCM-KNN				**96.13**	**96.10**	**96.13**	**3.45s**
DSIFT+GIST-KNN				**93.56**	**96.22**	**93.56**	**10.1s**
Zhihua et al. [22]	2018			92.10	91.70	91.90	60ms
				92.5	91.4	92.20	64ms
				92.70	92.30	92.50	48ms
				93.40	93.00	93.20	48ms
Aziz et al. [19]		128×128		NS	NS	89.11	NS
Songqing et al. [29]	2017	NS	Deep Learning	NS	NS	97.32	NS
Abien et al. [30]		NS		85	85	84.92	11m
Zhihua et al. [22]	2018	192×192		94.60	94.50	94.50	20ms
Our Method	-	**224×224**		**98.47**	**98.47**	**98.47%**	**17s**
		229×229		**98.79**	**98.79**	**98.79**	**30s**

Note: *CA=Classification Accuracy, PR=Precision Rate, RR=Recall Rate, NS=Not Specified*

5. Conclusion

The identification of malware attacks in the industrial-based Internet of Things is a key cyber security problem. Therefore we propose an architecture in this paper for detecting malicious activities in the IIoT environment. It uses a methodology that integrates malware visualization into the DCNN model. First of all, our malware visualization technique converts APK files to

color images. Next, a DCNN model is selected to extract the malware's dynamic image features. The DCNN model is used to train images of malware and detect the malware attacks. Finally, the accuracy of the detection of state-of - the-art methods is compared with the method proposed for challenging data sets. The detection accuracy on IIOT dataset of the proposed malware detection model was 97.81 % and Windows dataset was 98.47 %, respectively.

Most traditional malware detection techniques cannot be used directly for IIoT devices due to their computing complexity and limited resources, i.e. cost, memory and limited processing capabilities. Therefore, in the future, we will try to develop a combined blockchain and machine learning memory less malware detection model.

References

[1] Hamad N. Detection of Malicious Activities in Internet of Things Environment Based on Visualization Images and Machine Intelligence, Wireless Personal Communications, Springer, 2019,pp.1-21.

[2] Asaf S, Robert M, Yuval E, Chanan G. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey, Information Security Technical Report, 2009; 14(1):16-29.

[3] Manuel E, Theodoor S, Engin K and Christopher K. A Survey on Automated Dynamic Malware-Analysis Techniques and Tools, ACM Transaction, 2012; 44(2): 1-42.

[4] IDAPro. https://www.hexrays.com/products/ida/support/download_freeware.shtml

[5] OllyDbg. http://www.ollydbg.de/

[6] Ekta G, Divya B and Sanjeev S. Malware Analysis and Classification: A Survey, Journal of Information Security, 2014; 5: 56-64.

[7] Moser A, Kruegel C and Kirda E. Limits of Static Analysis for Malware Detection. In proceeding of 2007 Conference on Annual Computer Security Applications, 2007; 421-430.

[8] Eul G I ,KyoungSoo H, Jae H L, Boojoong K. Malware analysis using visualized images and entropy graphs, International Journal of Information Security, 2014; 1-14.

[9] KyoungSoo H, Jae H L, Boojoong K, Eul G I. Malware analysis using visualized image matrices, The Scientific World Journal, 2014; 1-15.

[10] Lakshman N S, Karthikeyan G J, Manjunath B S. Malware images: visualization and automatic classification, In proceeding of 2011 ACM Conference on Visualization for Cyber Security, 2011; 1-4.

[11] Ban X, Chen L, Hu W , Wu Q. Malware Variant Detection Using Similarity Search over Content Fingerprint, In proceeding of 2014 IEEE Conference on Control and Decision,2014; 5334-5339.

[12] Oliva A, Torralba A. Modeling the shape of the scene: a holistic representation of the spatial envelope, International Journal of Computer Vision, 2001, 145–175.

[13]Barath N N, Ouboti D B, Temesguen M K. Pattern Recognition Algorithms for Malware Classification, In proceeding of 2016 IEEE Conference of Aerospace and Electronics, 2016; 338-342.

[14] Kesav K, John D, Srinivas M. Packer identification using Byte plot and Markov plot, Journal of Computer Hacking Virology Techniques, 2016; 101-111.

[15] Kesav K, Srinivas M. Image Visualization based Malware Detection, In proceeding of 2013 IEEE Conference on Computational Intelligence in Cyber Security, 2013; 40-44.

[16] Bay H, Ess A, Tuytelaars T, and Van G L. Computer vision and image understanding, Speeded-Up Robust Features (SURF), 2008, 346–359.

[17] Lowe D. Object recognition from local scale-invariant features, ICCV, 1999;1150–1157.

[18] Jae H L ,KyoungSoo H, Eul G I., Malware Analysis Method using Visualization of Binary Files In proceeding of 2013 ACM Conference on Research in Adaptive and Convergent Systems, 2013; 317-321.

[19] Aziz M, Anita P. Malware Class Recognition Using Image Processing Techniques, In proceeding of 2017 IEEE Conference on Data Management, Analytics and Innovation, 2017; 76-80.

[20]Mahmoud K, Mrigank R, Noman M, Neil D B, Yang W, Farkhund I. Malware Classification with Deep Convolutional Neural Networks, In proceeding of 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018; 1-5.

[21] Rajesh K, Zhang X, Riaz U K, Ijaz A, Jay K. Malicious Code Detection based on Image Processing Using Deep Learning, in proceeding of International Conference on Computing and Artificial Intelligence (ICCAI), 2018; 81-85.

[22]Zhihua C, Fei X, Xingjuan C, Yang C, Gai-ge W, Jinjun C. Detection of Malicious Code Variants Based on Deep Learning, IEEE Transactions on Industrial Informatics, 2018; 3187-3196.

[23]ELENKOV N. Android Security Internals: An In-Depth Guide to Android's Security Architecture. No Starch Press, 2014.

[24]Bouvrie J. Notes on Convolutional Neural Networks. Technical Notes, 2006.

[25]Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. Technical Report, 2014: 1409-1556.

[26]Yao D, Junfeng W, Qing L. An Android Malware Detection Approach Using Community Structures of Weighted Function Call Graphs [J]. IEEE Access, 2017: 17478-17486.

[27]Zongqu Z, Junfeng W, Jinrong B. Malware Detection Method Based on the Control-Flow Construct Feature of Software [J]. IET Information Security, 2013: 18-24.

[28]Hashemi H, Hamzeh A. Visual Malware Detection Using Local Malicious Pattern, Journal of Computer Virology and Hacking Techniques, 2018:1–14.

[29]Songqing Y. Imbalanced Malware Images Classification: a CNN based Approach. Cornell University Library, 2017: 1-6.

[30]Abien F M, Francis J H P. Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach Using Support Vector Machine (SVM) for Malware Classification, Technical Report, 2017: 1-5.

[31]Hamad N, Bing G, Danish V, Rashid M. N, Farhan U, Hamza A et al. Identification of malicious code variants based on image visualization, Journal of Computers and Electrical Engineering, Elsevier, 2019:225-237.

[32]Hamad N, Bing G, Farhan U, Rashid M. N. A Cross-Platform Malware Variant Classification based on Image Representation, KSII Transactions on Internet and Information Systems, Korean Internet Society, 2019:3756-3777.

[33]Muhammad R. N, Tao L, Hamad N, Farhan U and Saqib S, "Scalable Mutation Testing Using Predictive Analysis of Deep Learning Model," IEEE Access, 2019:158264-158283.

[34]Farhan U, Hamad N, Sohail J, Shehzad K, Muhammad A. L, Fadi A, Leonardo M. "Cyber Security Threats detection in Internet of Things using Deep Learning approach", IEEE Access, 2019:124379-124389.

[35]Nasir N. H, Shabir A. P,Javaid A. S,Fadi A,Khan M. “Secure Data Transmission Framework for Confidentiality in IoTs”, Elsevier Ad Hoc Networks Journal, 2019.

[36]Al-Turjman F, Zahmatkesh H, Shahroze R. "Smart-cities Medium Access for Smart Mobility Applications in IoT", Wiley Transactions on Emerging Telecommunications Technologies, 2019.

[37]Dr. B. D. Deebak, Fadi Al. “A Hybrid Secure Routing and Monitoring Mechanism in IoT-based Wireless Sensor Networks”, Elsevier Ad Hoc Networks Journal, 2020.

[38]Fadi A, Hadi Z, Leonardo M. “Quantifying Uncertainty in Internet of Medical Things and Big-Data Services Using Intelligence and Deep Learning”, IEEE Access, 2019:115749-115759.