EasyChair Preprint

№ 641

Opcode and Gray Scale Techniques for Classification of Malware Binaries

Rajesh Kumar and Riaz Ullah Khan

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 19, 2018

Opcode and Gray Scale Techniques for Classification of Malware Binaries

Rajesh Kumar, Riaz Ullah Khan

Center of Cyber Security,
School of Computer Science & Engineering,
University of Electronic Science and Technology of China.
e-mail: rajakumarlohano@gmail.com, riazkhan@ieee,org

***Abstract*—In this study, we have used the image similarity technique to detect the unknown or new type of malware using CNN approach. CNN was investigated and tested with three types of datasets i.e. one from Vision Research Lab, which contains 9458 gray-scale images that have been extracted from the same number of malware samples that come from 25 different malware families, second was from Microsoft Malware Classification Challenge which contains 10868 Binary files that contains 9 different malware families and third was benign dataset which contained 3000 different kinds of benign software. Benign dataset and dataset from Microsoft Malware Classification Challenge were initially .EXE files which were converted into binary code and then converted into image files. We obtained a testing accuracy of 98% on Vision Research lab dataset and 97.6 accuracy on Microsoft Malware Classification Challenge dataset.**

***Index Terms*—Malware Detection, Convolutional Neural Network, Malware Classification, Deep Learning**

I. Introduction

A. Background

One of the major challenges in the realm of security threats is malicious software which is also referred as malware. The main focus of malware is, to gather the personal information without the attention of users and to disturb the computer operations which makes problems for users. There are many kinds of malware i.e. Virus, Worm, Trojan-horse, Rootkit, Backdoor, Spyware, Adware etc. Annual reports from antivirus companies show that thousands of new malware are created every single day. These new malware become more sophisticated that they could no longer be detected by the traditional detection techniques such as signature-based detection, heuristic detection or behavior-based detection.

Signature-based detection searches for specified bytes sequences into an object so that it can identify exceptionally a particular type of a malware. Its drawback is that it cannot detect zero-day or new malware since these malware signatures are not supposed to be listed into the signature database. Heuristic-based detection was developed to basically overcome the limitation of the signature detection technique, in the way that it scans the system's behavior in order to identify the activities which seems to be not normal, instead of searching for the malware signature. Heuristic-based detection method can be applied to newly created malware whose signature has not yet been known. The limitation of this technique is that it affects the system's performance and requires more space. Behavior-based detection technique is more about the behavior of the program when it is executing. If a program executes normally, then it is marked as benign, otherwise it is marked as a malware. By analyzing this definition of the behavior-based detection, we can directly conclude that the drawback of this technique is the production of many false positives and false negatives, considering the fact that a benign program can crashed and be marked as a virus or virus can execute as if it was a normal program and simply be marked as benign.

B. Motivations

Malware is growing in the huge volume every day, we used image processing technique in order to improve accuracy and performance. Image processing technique analyzes malware binaries as gray-scale images. The previous research [29] proposed a new method for visualization to classify malware using image processing technique. Some of mature image processing techniques are widely used for object recognition e.g. taobao is popular shopping website in china which find's the product using image recognition technique. This method performs high accuracy in practice. In this study, we converted binary code to images for recognizing malware which preserve the similarities variant images. We observed that the image recognition method is helpful to achieve better performance and accuracy.

C. Our approach

Malware classified in different families has multiple characteristics or features. Many authors used machine learning models such as Regression, K-nearest-neighbor, Random Forest etc. Main disadvantage of using machine learning is, features extraction is manual. Gavrilut et al. [15] gave an overview of different machine learning techniques that were previously proposed for malware detection. Unlike Machine Learning, Deep learning skips the manual steps of extracting features. For instance, we can feed directly images and videos to the deep learning algorithm, which can predict the object. In this way deep learning model is more intelligent rather than machine learning model. We used convolutional neural networks because it is reliable and it can be applied to the entire image at a time and then we can assume they are best to use for feature extraction. Recently Constitutional Neural Networks [8] is the new approach to detect malware by using image based similarity technique. Its automated image comparison helps analysts to visually identify common code portions or specific instruction blocks within a sample. In this work we used three

different datasets and compared the accuracy. Secondly we used different techniques to prepare datasets for training and testing purposes. we trained and tested the CNN model for better understanding of the malware behavior. Overall, we show that our proposed approach constitutes a valuable asset in the fight against malware.

D. Contributions

The main contributions of the paper are summarized as follows;

- We used the Constitutional Neural Networks for detection of malware, based on image similarity which is further described in Section V
- We compared gray scale image and opcode sequence accuracy.
- We successfully analyzed and detected unknown or new type of malware
- We build opcode sequence algorithm which takes less time to train the classifier.
- We achieved better results in terms of training / testing accuracy and speed of detection which is further described in section V
- We collect different malware datasets from different sources in section IV and compare the accuracy.
- We achieved 98% of accuracy on Vision Research Lab's Dataset and 97.6% accuracy on Microsoft Malware Classification Challenge dataset on gray scale image construction.
- We achieved 87.98% accuracy on Microsoft Malware Classification Challenge dataset on opcode image construction.

E. Structure of paper

The Section I discusses the background, motivation, approach used in this study and main contributions of this work. Section II discusses problem description. Section III briefly discusses the three detection techniques (Signature, Heuristic, Cloud) and four analysis techniques (Static, Dynamic, Statistical and Content Analysis, Hybrid Analysis). Section IV gives a brief overview to the methodology that how to convert executable files in to images and also setup the python libraries. Section V proposes a malware detection technique, discusses optimized CNN model, describes the implementation and experiment results in terms of accuracy. Section VI gives a brief discussion on previous work done in similar field. Finally, Section VII concludes the paper.

II. Problem Description

In this study, we tried to solve the problem of malware detection by using deep learning algorithm which provide more accuracy and speed. In table I, researchers used malware detection techniques using machine learning algorithms. In our work, we tried to achieve more accuracy for the same deep learning model with different datasets.

Table I
Different Machine Learning Techniques used for Malware Detection

Author	Goal	System components	ML Techinques
Ahmed[3]	Detect malware by Windows API call traces	Kernel mode hook, knowledge base, API call trace, feature extractor, training, classifier	C4.5 decision tree, instance-based KNN, NB, inductive rule learner, SVM-SMO
Kolbitsch [23]	Detect malware at end hosts	Knowledge base, program slicing, behavioral profile generation, matching	Directed acyclic graph(DAG) matching algorithms
Lanzi [17]	Diversity of system calls study (System-centric malware analysis)	Training dataset, system call sequence miner	N-gram models
Ye[33]	Detect malware by file relations	File relation and content collector, feature extractor, classifier	Customized parametric model on content and non-parametric one on relations
Antonakakis Sec'12[114]	Detect domain generation algorithm (DGA)-based malware	Knowledge base, discovery engine, trainer, classifier within networks	X-means clustering, spectral clustering, decision tree, hidden Markov model(HMM)
Rahman [30]	Detect malware propagated by social network	User authorization, post crawler, feature extractor, WL, BL, train(manually labeled data) classifier, user feedback	SVM (kernel unspecified)
Tamersoy [31]	Detect malware by file relation graphs	File collector, LSH, graph builder, belief propagator	MinHash, LSH, pairwise Markov random field, unweighted bipartite graph
Invernizzi [18]	Detect malware downloads in networks	Network traffic collector, feature extractor, distributed classifier	Decision tree(ground truth)
Arp [6]	Explainable Android malware detection	Broad static analysis, feature embedder, detector, explanation	Linear SVM
Graziano [17]	Detect and forecast malware samples and trends from public dynamic analysis sandbox	Dynamic analysis, binary similarity, fine-grained static analysis, classifier	Logistic model tree
Kinder [22]	Proactive detection of computer worms using model checking	control flow graph from the binary and automatically verifies it against a formal malware specification	language CTPL
Alazab[4], [5]	Detect Malware using opcode frequency and classification of malwares	Analysis Malwre uisng opcode frequency	

III. Malware Detection and Analysis Techniques

A. Malware Detection Techniques

1) Signature Based Malware Detection: A Signature is a short sequence of unique bytes, also it identified the unique string from the binary code [28]. The process of the traditional detection method shown in figure 1.

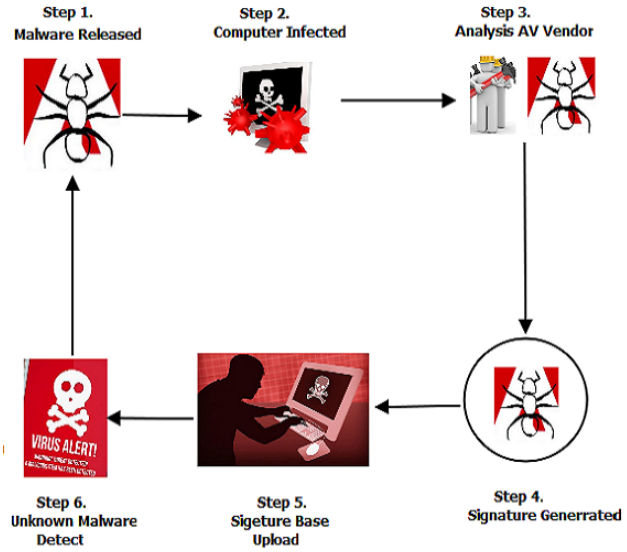


Figure 1. Signature Based Malware Detection

2) Heuristic Based Malware Detection: It is based on rules/ patterns and these rules based on generic enough to be consistent with alternatives of the same malware threat. But these rules are not useful for benign files [33].

3) Cloud Based Malware Detection: Now a days many anti malware vendors used the cloud based detection, because antivirus software unable to detect many modern malware threats. Advantage of cloud based detection is, it is more secure because it detects malicious code by using multiple detection engines.

B. Malware Analysis

1) Static Analysis: Analysis of the bad code segment or malicious characteristics when code is not executing is called static analysis [7], [11], [12]. There are many patterns to detect the static malwares e.g. n-grams, string signature, control flow graph, bytes-sequence etc.

2) Dynamic Analysis: Analysis of the bad code or malicious characters when software is running environment i.e. emulator,Virtual Machine, simulator etc. is called dynamic analysis. Dynamic analysis approach applied for monitoring and tracing system. Dynamic analysis is better approach rather than static analysis but it consumes more resources i.e. time and memory and it is also scalability issue.

3) Statistical and Content Analysis: This technique is based on verity of techniques e.g. n-gram, n-perms, hash based, file structure.

4) Hybrid Analysis: Analysis of the bad code or malicious characters while performing static and dynamic analysis in an offline mode [32], [2]. It is also useful for android application. In the first step of this method, the static analysis take the image of the applications into small pieces which is also known as binary code and search suspicious patterns among the binary codes. In the second step, the dynamic analysis executes the code in an Android emulator and logs its system calls.

IV. Data Preparation and Environment Setup

This section is divided into two parts. The first part is, to collect malware and benign datasets from different sources and second part describes the techniques of preparation of the dataset. In second part we used a technique to prepare dataset which is described in Section IV-B.

A. Collection of Dataset

We have collected three datasets from different sources. Two of them are malicious datasets from two different sources i.e. from Vision Research Lab and from Microsoft Malware Classification Challenge. We also collected 3000 benign file from different sources. All three datasets are discussed briefly in the following discussions.

1) Vision Research Lab Dataset: First dataset is collected from Vision Research Lab and this dataset is called Malimg Dataset [29]. The dataset comprises 25 malware families while the number of variants is different in each family. Dataset is shown in Table II along with class name, family name and number of samples.

Table II
Malimg Dataset from Vision Research Lab Dataset

No	Class	Family Name	No of Samples
1	Worm	Allaple.L	1591
2	Worm	Allaple.A	2949
3	Worm	Yuner.A	800
4	PWS	Lolyda.AA 1	231
5	PWS	Lolyda.AA 2	184
6	PWS	Lolyda.AA 3	123
7	Trojan	C2Lop.P	146
8	Trojan	C2Lop.gen!G	200
9	Dialer	Instantaccess	431
10	Trojan Downloader	Swizzor.gen!I	132
11	Trojan Downloader	Swizzor.gen!E	128
12	Worm	VB.AT	408
13	Rogue	Fakerean	381
14	Trojan	Aluron.gen!J	198
15	Trojan	Malex.gen!J	136
16	PWS	Lolyda.AT	159
17	Dialer	Adialer.C	125
18	Trojan Downloader	Wintrim.BX	97
19	Dialer	Dialplatform.B	177
20	Trojan Downloader	Dontovo.A	162
21	Trojan Downloader	Obfuscator.AD	142
22	Backdoor	Agent.FYI	116
23	Worm:AutoIT	Autorun.K	106
24	Backdoor	Rbot!gen	158
25	Trojan	Skintrim.N	80

Malimg Dataset consists 9,458 gray-scale images of 25 malware families. Ratio of 90-10 was used for model performance evaluation. 90% of the total data was used for training and 10%

was used for testing. The real malware binaries of this dataset was available in [1],

As Gavrilut et al. [16] explained that a binary code of a given malware can be read as a vector of 8 bits un-signed integers and organized into 2-dimensional array which can be visualized as a gray-scale image in the range of [0,255], where 0 represent black and 255 for white. The size of the image is different depending on their families. We observed in Figures 2, that images which belong to the same family are looking very similar to one another.

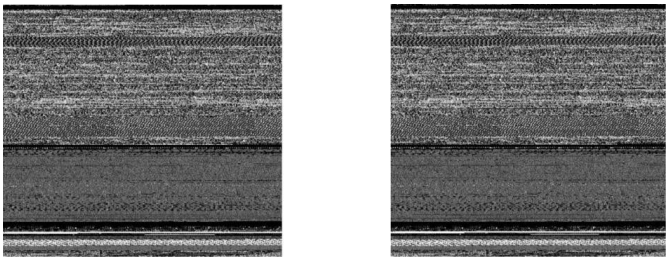


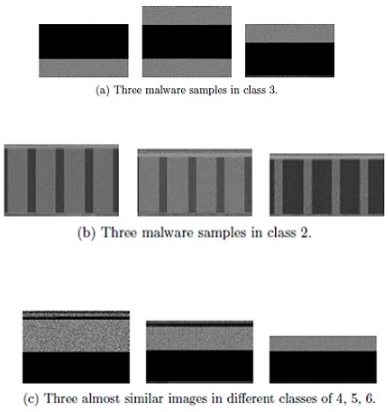
Figure 2. Images extracted from malware binaries

2) Microsoft Malware Classification Challenge Dataset: The Microsoft dataset contains 9 classes for training and testing purposes. Microsoft provided 500GB of data which includes 21741 malware samples. 10868 of samples are used for training, and the remaining samples are used for testing.

1) **Bytes Files:** Byte files in Microsoft dataset include 10,868 training data and 10873 testing data. Each byte file contain hexadecimal representation of binary content.
2) **Asm Files:** Asm files in Microsoft dataset include 10,868 training data and 10873 tasting data. Each asm file extracted by the IDA dis-assembler tool and it contains metadata manifest. This information includes assembly command sequences, strings, function calls and so on.
3) **Training Labels:** MD5 Hash is the file name in actual program and this name is used as a training label. The file of training label contains each MD5 hash and class of malware which it maps to. No training labels were provided for the test data input files.
4) **Sample Submission:** The sample submission file illustrates the valid submission format for 10,873 sample records.
5) **Data Sample:** The data sample file includes a preview of the test and training data.

Table III
MICROSOFT MALWARE CLASSIFICATION CHALLENGE DATASET

No	Family Name	No of Samples
1	Ramnit	1541
2	Lollipop	2478
3	Kelihos_ver3	2942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator.ACY	1128
9	Gatak	1013



(a) Three malware samples in class 3.

(b) Three malware samples in class 2.

(c) Three almost similar images in different classes of 4, 5, 6.

Figure 3. Images extracted from malware Microsoft dataset

3) Benign files : We collect 3000 benign files from different sources.

B. Data Preparation Techniques

This paper propose the following two technique to process the data.

1) Direct Convert Assembly To Image:

1) Decompiling: we used the following algorithm to decompile the exe file to binary and assembly.
2) Convert Assembly Code to Image: We converted the assembly code to image. The process of converting assembly to images.

2) Opcode To Image:

1) Unpacking: In this method we use two tools for extracting hidden code from binary files
   a) Themida and vmprotect
   b) UPX
2) Decompiling: In this phase we decompile opcode sequence from assembly code and then convert 2-tuple opcode sequence rather than larger length of opcode sequence.
3) Opcode Sequence: The binary image matrices are reconstructed by these opcode sequences with \their probabilities and information gains.The matrix is shown in Figure 2, each opcodes sequences of length 2 can be matched to one of the elements in the matrix according to $os_i =< op_j, op_k >$,as shown in def 3 , def.. The element value $val(os_i \mid x_j)$ of the image matrix $im(x_j)$ is calculated by the probabilities $p(os_i \mid x_j)$ and the information gains $w(os_j)$ of os_iin binary x

$$val(os_i|x_j) = p(os_i|x_j)w(os_j) \tag{1}$$

The probabilities $p(os_i \mid x_j)$ and information gains $w(os_i)$ are calculated by the frequencies $freq(os_i \mid x_j)$ of the opcodes sequences of length 2 , as shown in Eq 2 and 3, where $p(os_i \mid y_1)$ be the probability of os_iin the training malware binaries, $p(os_i)$ be the probability of os_i in the whole training binaries, and $p(y_1)$ be the probability of training malware binaries.

$$p(os_i|x_j) = \frac{freq(os_i|x_j)}{\sum_{os_t Ex_j} freq(os_i|x_j)} \tag{2}$$

$$w(os_i) = p(os_i|x_j) \log\left(\frac{p(os_i|x_j)}{p(os_i)p(y_i)}\right) \quad (3)$$

1) Binary Image Re-construction and Enhancement: Images are constructed by binary opcode frequency. Opcode sequences will be enhanced by using histogram normalization, dilation, and erosion techniques

To enhance the contrast between malware variant images and benign images, we use histogram normalization. dilation and erosion methods to enhance the binary images. Through image enhancement, the contrast of these special opcode images would be enhanced,

Let $val_{enhance}(os_i \mid x_j)$ be the pixel-value of the enhanced image, the histogram normalization method is according to the eq 4

$$val_{enhance}(os_i|x_j) = \alpha \frac{val(os_i|x_j)}{max(val(os_i|x_j))} 255 \quad (4)$$

C. Environment Setup

CentOS system with 64bit with 64 GB RAM environment is used to perform tests. We used Python programming language to perform the experiments. Python packages and libraries such as Tensor Flow, Docker Server, Anaconda are used which helped to detect the malware. The Tensor Flow Library is used for training the model which uses the convolutional natural network (CNN).

V. Implementation and Performance Evaluation of the Proposed Model

A. Proposed Model

In this design we divided model in two phases i) Training phase and ii) Detection phase. For the training and the detection of malware we used CNN model. We prepare the dataset using different techniques shown in data preparation section. The output of the data preparation section is "image files". Images have binary labels i.e. either benign or malware. we used supervised learning model in which the features are extracted automatically. The same exe file convert in image and trained classifier detect the malicious code.

B. Training Convolutional Neural Networks Structure

We have used convolutional neural networks because it is reliable and it can be applied to the entire image at a time and then we can assume they are best to use for feature extraction. convolutional neural network is a feed-forward neural network where the connectivity pattern between neurons is inspired by the structure of an animal visual cortex and that has proven great value in the analysis of visual imagery.

Firstly we used auto-encoders scheme , auto-encoders is widely used for dimensionality reduction and data de-nosing.

$$f(x) = g(\sum a_i x_i + c) \quad (5)$$

Where $g^{\bigotimes}$ defines a linear function as an independent variable function.

$$h^{(i)} = g^{\otimes}(w^{(i)^T} x + b^{(i)}) \quad (6)$$

Usually we want function g to be a non-linear function, and we also need it to be easily derived. Therefore, we generally use ReLU (Linear Rectifier) function g (z) = max (0, z). Other types of activation functions g also include logistic functions:

$$g(z) = \frac{1}{1 + e^{-2\beta z}} \quad (7)$$

For example, the hypeFor example, the hyperbolic tangent function:bolic tangent function:

$$g(z) = \tanh(z)\frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (8)$$

Some inhibitory functions are also used in the competitive neural network, such as:

$$map_j(x_i) = pool(g(\sum conv(w^{(1)} {map_j}^{(1)}(x_i))) \quad (9)$$

Convolution neural networks have a wide range of applications in image processing. In the convolution neural network, in order to simplify the number of parameters of the neural network, the convolution neural network uses the weight sharing and the convolution kernel mechanism. For example, for a three layer convolution neural network:

$$y = \frac{e^{-u}}{\sum e^{-u}} \quad (10)$$

$$w^{(3)} = w^{(3)} + c.d_{output}.{map_j}^{(2)}(x^i) \quad (11)$$

$$d_{output} = (y - y^{(3)}).y^{(3)} \quad (12)$$

$$d^{(3)} = y^{(3)}(1 - y^{(3)}).\sum d_{output}.w^{(3)} \quad (13)$$

$$w^{(2)} = w^{(3)} + c.d^{(2)}.conv({map_j}^{(1)}(x_i)) \quad (14)$$

$$w^{(1)} = w^{(1)} + c.d^{(1)}.conv(img(x_i)) \quad (15)$$

The advantage of eq (7) and (8) these two activation functions relative to ReLU are that they are both bounded functions.

In this method of data preparation, we can easily identify malware and benign files by visual analysis as shown in Figure **??**.

All the images are reshaped into a size of 128 X 128 pixels. Since all the models of deep learning accept data in form of numbers, we have used image library from PIL package of Python to generate vectors of images and further processing are done on these vectors.

We have then designed a three layers deep Convolutional Neural Network for the detection task, which has the following properties: On the Rectified Linear Units (ReLU) layers, we first apply a two dimensional convolutional layer and after each layer, we applied a nonlinear later also known as activation layer. In convolutional layer, we have operations like

element-wise multiplication and summations. The ReLU adds non-linearity to the system. We have used the ReLU instead of non-linearity function because it is faster than tanh or sigmoid and help in vanishing gradient problem which arises in lower layers of the network.

We have also used max pooling layer instead of other layers. It takes a filter and a stride of the same length then applies it to the input volume and outputs the maximum number in sub region that the filter involves around. The intuition behind this was the fact that our malware image is a gray scale and the layers like average max pooling may not help much because there are a lot of dark space in the image and they don't contribute much in the model.

The output that we want is a single class in which the given malware belongs to. After applying all the layers, we have a three-dimensional vector of arrays. To convert this vector into a class probability, we convert these vectors into a single layer of one dimension, known as fully connected layer. Down-sampling all the vectors to a one-dimensional vector may lead to loss of data. For that reason, we have used two fully connected layers.

Cross entropy loss function that is commonly used for multi class classification was used for this work as well as Adam optimizer for optimization task. The overall architecture of the model is show in Figure 4.

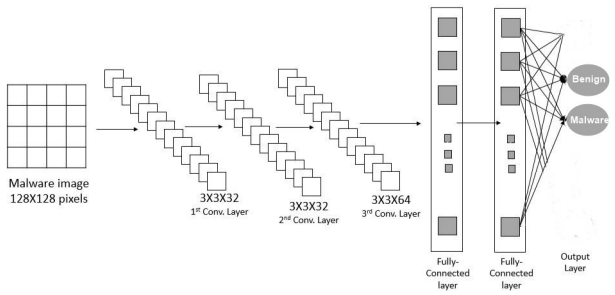


Figure 4. Overview architecture of CNN proposed Method

Initially, all the images were of different sizes and had to be converted into 128 X 128 pixels before they are used as input to the model.

C. Implementation

The following tools and techniques are required for experimental setup. For preparation of dataset we used method shown in section IV, tools and algorithm which were used to arrange the dataset for achieving better results is also written in data preparation section. For detecting malware we use supervised learning to train the model. CNN algorithm was used to train and test the model.We used 3 hidden layers, each layer has own parameters (e.g., $filter - size1 = 3$, $num_filters = 32$, etc), In this algorithm we used $AdamOptimizer$ and the learning rate of the optimizer is le-4. The size of for all hidden layers for the convolutional neural network are 3*3*32, 3*3*32, 3*3*64, respectively. For the validation, system was trained with 20 epochs.

D. Experiment Results

We took three datasets in considerations which are discussed in Section IV. Two of the three datasets consist of malicious code and one dataset is a benign file. For the first test, we combined the benign dataset with Malimg dataset and used the combined dataset to obtain the accuracy in terms of malware code detection. For the second test, we combined the same benign dataset with Microsoft dataset and obtained the accuracy in terms of malware code detection.

In this Experiment we obtain accuracy from different methods and compare both methods one opcode sequence and other gray-scale image

- The result obtained from gray scale image shows an accuracy of 98% for the Dataset of Vision Research Lab shown in Figure 6.
- The result obtained form gray scale image achieved 97.6 % accuracy on Microsoft Malware Classification Challenge Dataset.
- The result obtained form opcode image achieved 87.98% accuracy on training dataset and 88.36 on validation dataset on Microsoft Malware Classification Challenge Dataset shown in Figure 5.
- The training time of opcode and gray scale is 5580 and 62238 seconds.

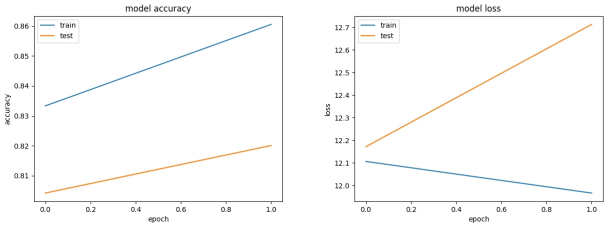


Figure 5. Model accuracy and model loss of the Microsoft Malware Classification Challenge Dataset

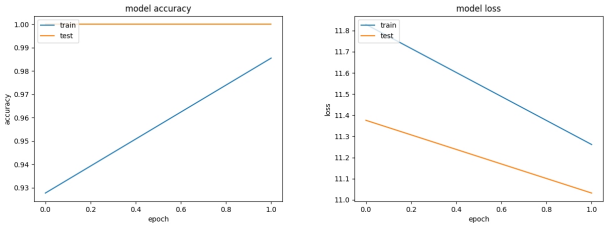


Figure 6. Model accuracy and model loss of the Vision Research Lab Dataset

VI. Related Work

Various machine learning algorithms are proposed for classification and detecting unknown codes in to either their unknown families for instance Naive Bayes, Support Vector Machine, Clustering, and Association Rule. In this section, we will discuss few of the good researchers who have worked in the field of malware detection and classification. Khan et. al. [20], [21] analysed ResNet and GoogleNet models for malware detection which are based on CNN. Kumar et. al. [25] used CNN model for malicious code detection based on

pattern recognition. Egele et al. [10] analyses the behavior of malware. They have designed the binary obfuscation methods, which transform the malware binaries in to self-compressed. They have also designed a technique that is uniquely identify binary files which restricted the reverse engineering.

Nataraj et al. [29] used image processing technique to classify the malware. They converted binary malware to gray-scale images. The proposed method of Nataraj et al. [29] represents executable binary files into gray-scale bitmap images. Kong et. al. [24] built a model to classify the malware, based on structural information. For the structural information, they use the function call graph, this function extracts the features of each malware sample. they used the discriminate distance metric learning method which cluster the malware samples belongs to same family and also used assemble of classifier that classify malware into their respective families.

Firdausi et al.[13] Used the machine learning algorithms i.e., K-Nearest Neighbors, SVM, Naive Bayes, J48 Decision Tree and Multi-layer Perceptron Neural Network for malware detection. The performance compares by 5 different algorithms and the obtained best performance achieved by j48 decision tree. They take small amount of sample 220 malicious and 250 benign, the obtained results achieved 96.8 % accuracy. Gandotra et al. [14] publish a survey paper for the malware classification and detection, they describe many machine learning algorithm technique for the detection of malware but in this contribution nobody use deep learning models(e.g., Convolutional Neural Network (CNN) to detect the malware. Damshenas et. al. [9] proposed a technique for detecting malware in mobile devices. This technique is comprising a server analyzer and a lightweight client agent. The server analyzer generates a signature for every application. The proposed technique is capable of generating standardized mobile malware signatures based on their behavior and this is the main contribution of their research. It compare the generated signature and previously blacklist of malware signature.

N. Milosevic et al.[27] proposed the static analysis approach to detect and analyse malicious behavior within the code in android apps. They use the machine learning approach to detect the malware families, this it is also signature based anti-malware solution. They used Service Vector Machine (SVM) for finding the accuracy, the results were 95.6 percent. Lee et. al. [26] also used machine learning to solve the problem of signature generating for worm. They use the nearest neighbor technique for cluster of malicious programs. Tain at al. [19] focused in classify Trojans. They use function length frequency, the amount of bytes that determine function length in the cipher text. Their model's performance show that the range of function along with its frequency are meaningful to identify the malware families.

VII. CONCLUSION

Being able to visualize the malicious code as an image has been a great achievement. Many researchers have been using this technique for the task of malware classification and detection. However, other works have shown that this technique can be easily vulnerable to adversarial attacks and produce erroneous results. This was observed that how a small change in the image could lead to miss-classification of images. The biggest challenge is to find an efficient way to overcome the vulnerability of Neural Networks. This could be achieved by carefully analyzing malware binaries.

REFERENCES

[1] *Vision Reseach Lab Malimg Dataset http://old.vision.ece.ucsb.edu/spam/malimg.shtml*.

[2] F Afifi, N B Anuar, S Shamshirband, and K K R Choo. DyHAP: Dynamic Hybrid ANFIS-PSO Approach for Predicting Mobile Malware. *PloS one*, 2016.

[3] F Ahmed, H Hameed, and M Z Shafiq. Using spatio-temporal information in API calls with machine learning algorithms for malware detection. *Proceedings of the 2nd*, 2009.

[4] Mamoun Alazab. Profiling and classifying the behavior of malicious codes. *Journal of Systems and Software*, 100:91–102, 2015.

[5] Mamoun Alazab, Mohammad Al Kadiri, Sitalakshmi Venkatraman, and Ameer Al-Nemrat. Malicious code detection using penalized splines on opcode frequency. In *Cybercrime and Trustworthy Computing Workshop (CTC), 2012 Third*, pages 38–47. IEEE, 2012.

[6] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In *Proceedings 2014 Network and Distributed System Security Symposium*, 2014.

[7] D Barrera, H G Kayacik, and P C van Oorschot. A methodology for empirical analysis of permission-based security models and its application to android. *Proceedings of the 17th*, 2010.

[8] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. pages 2956–2964, 2015.

[9] Mohsen Damshenas, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Ramlan Mahmud. M0Droid: An Android Behavioral-Based Malware Detection Model. *Journal of Information Privacy and Security*, 11(3):141–157, 2015.

[10] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, 44(2):6, 2012.

[11] William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*, page 235, 2009.

[12] AP Felt, Erika Chin, and Steve Hanna. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11 (2011)*, pages 627 – 636, 2011.

[13] I Firdausi, A Erwin, and A S Nugroho. Analysis of machine learning techniques used in behavior-based malware detection. *Advances in Computing,*, 2010.

[14] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware Analysis and Classification: A Survey. *Journal of Information Security*, 05(02):56–64, 2014.

[15] Dragos Gavrilut, Mihai Cimpoesu, Dan Anton, and Liviu Ciortuz. Malware detection using machine learning. In *2009 International Multiconference on Computer Science and Information Technology*, pages 735–741. IEEE, oct 2009.

[16] Dragoş Gavriluţ, Mihai Cimpoeşu, Dan Anton, and Liviu Ciortuz. Malware detection using machine learning. In *Computer Science and Information Technology, 2009. IMCSIT'09. International Multiconference on*, pages 735–741. IEEE, 2009.

[17] M Graziano, D Canali, L Bilge, and A Lanzi. Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence. *USENIX Security*, 2015.

[18] L Invernizzi, S Miskovic, R Torres, C Kruegel, and S Saha. Nazca: Detecting Malware Distribution in Large-Scale Networks. *NDSS*, 2014.

[19] R Islam, R Tian, L M Batten, and S Versteeg. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer*, 2013.

[20] Riaz Ullah Khan, Xiaosong Zhang, and Rajesh Kumar. Analysis of resnet and googlenet models for malware detection. *Journal of Computer Virology and Hacking Techniques*, Aug 2018.

[21] Riaz Ullah Khan, Xiaosong Zhang, Rajesh Kumar, and Emelia Opoku Aboagye. Evaluating the performance of resnet model based on image recognition. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, ICCAI 2018, pages 86–90, New York, NY, USA, 2018. ACM.

[22] Johannes Kinder, Stefan Katzenbeisser, Christian Schallhart, and Helmut Veith. Proactive detection of computer worms using model checking. *IEEE transactions on dependable and secure computing*, 7(4):424–438, 2010.

[23] C Kolbitsch, P M Comparetti, and C Kruegel. Effective and Efficient Malware Detection at the End Host. *USENIX security*, 2009.

[24] Deguang Kong and Guanhua Yan. Discriminant malware distance learning on structural information for automated malware classification. pages 1357–1365, 2013.

[25] Rajesh Kumar, Zhang Xiaosong, Riaz Ullah Khan, Ijaz Ahad, and Jay Kumar. Malicious code detection based on image processing using deep learning. In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, ICCAI 2018, pages 81–85, New York, NY, USA, 2018. ACM.

[26] Hwan-Taek Lee, Dongjin Kim, Minkyu Park, and Seong-je Cho. Protecting data on android platform against privilege escalation attack. *International Journal of Computer Mathematics*, 93(2):401–414, feb 2016.

[27] N Milosevic, A Dehghantanha, and K K R Choo. Machine learning aided Android malware classification. *Computers {&} Electrical*, 2017.

[28] Robert Moskovitch, Dima Stopel, Clint Feher, Nir Nissim, Nathalie Japkowicz, and Yuval Elovici. Unknown malcode detection and the imbalance problem. *Journal in Computer Virology*, 5(4):295–308, 2009.

[29] Lakshmanan Nataraj, Vinod Yegneswaran, Phillip Porras, and Jian Zhang. A Comparative Assessment of Malware Classification Using Binary Texture Analysis and Dynamic Analysis. *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pages 21–30, 2011.

[30] M S Rahman, T K Huang, and H V Madhyastha. Efficient and Scalable Socware Detection in Online Social Networks. *USENIX security*, 2012.

[31] A Tamersoy, K Roundy, and D H Chau. Guilt by association: large scale malware detection by mining file-relation graphs. *Proceedings of the 20th ACM*, 2014.

[32] F Tong and Z Yan. A hybrid approach of mobile malware detection in Android. *Journal of Parallel and Distributed Computing*, 2017.

[33] Y Ye, T Li, S Zhu, W Zhuang, E Tas, and U Gupta. Combining file content and file relations for cloud based malware detection. *Proceedings of the 17th*, 2011.